

EPICS-TwinCAT IOC

2.0

Generated by Doxygen 1.8.15

1 Module Index	1
1.1 Modules	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	11
5.1 File List	11
6 Module Documentation	13
6.1 Device support for TwinCAT/ADS	13
6.1.1 Detailed Description	16
6.1.2 Enumeration Type Documentation	16
6.1.2.1 epics_record_enum	16
6.1.3 Function Documentation	17
6.1.3.1 get_callback_queue_free()	17
6.1.3.2 get_callback_queue_size()	18
6.1.3.3 get_callback_queue_used()	18
6.1.3.4 linkRecord()	19
6.1.3.5 register_devsup()	20
6.1.4 Variable Documentation	20
6.1.4.1 isCallback	20
6.1.4.2 isPassive	20
6.1.4.3 the_register_devsup	21
6.1.4.4 value_alt_type	21
6.1.4.5 value_count	21
6.2 Device driver functions	22
6.2.1 Detailed Description	22
6.3 Info interface classes and functions	23
6.3.1 Detailed Description	23
6.3.2 Typedef Documentation	23
6.3.2.1 info_dbrecord_type	23
6.3.3 Enumeration Type Documentation	23
6.3.3.1 update_enum	23
6.4 TwinCAT tpy file parser	25
6.4.1 Detailed Description	25
6.4.2 Typedef Documentation	25
6.4.2.1 dimension	26

6.4.2.2 dimensions	26
6.4.2.3 enum_map	26
6.4.2.4 enum_pair	26
6.4.2.5 item_list	26
6.4.2.6 symbol_list	27
6.4.2.7 type_multimap	27
6.4.3 Enumeration Type Documentation	27
6.4.3.1 type_enum	27
6.5 XML tpy file constants	28
6.5.1 Detailed Description	29
6.6 Parser utility functions and classes	30
6.6.1 Detailed Description	30
6.6.2 Typedef Documentation	30
6.6.2.1 property_el	31
6.6.2.2 property_map	31
6.6.2.3 replacement_table	31
6.6.3 Enumeration Type Documentation	31
6.6.3.1 opc_enum	31
6.6.3.2 process_tag_enum	32
6.6.3.3 process_type_enum	32
6.7 OPC property constants	33
6.7.1 Detailed Description	34
6.7.2 Variable Documentation	34
6.7.2.1 OPC_PROP_ALIAS	34
6.7.2.2 OPC_PROP_ALMAREAS	34
6.7.2.3 OPC_PROP_ALMCONDITION	34
6.7.2.4 OPC_PROP_ALMCOSV	35
6.7.2.5 OPC_PROP_ALMDB	35
6.7.2.6 OPC_PROP_ALMDEV	35
6.7.2.7 OPC_PROP_ALMFFSV	35
6.7.2.8 OPC_PROP_ALMH	36
6.7.2.9 OPC_PROP_ALMHELP	36
6.7.2.10 OPC_PROP_ALMH	36
6.7.2.11 OPC_PROP_ALMHHSV	36
6.7.2.12 OPC_PROP_ALMHSV	37
6.7.2.13 OPC_PROP_ALML	37
6.7.2.14 OPC_PROP_ALMLIMIT	37
6.7.2.15 OPC_PROP_ALMLL	37
6.7.2.16 OPC_PROP_ALMLLSV	38
6.7.2.17 OPC_PROP_ALMLSV	38
6.7.2.18 OPC_PROP_ALMOSV	38
6.7.2.19 OPC_PROP_ALMPRIMARYAREA	38

6.7.2.20 OPC_PROP_ALMROC	38
6.7.2.21 OPC_PROP_ALMSTAT	39
6.7.2.22 OPC_PROP_ALMUNSV	39
6.7.2.23 OPC_PROP_ALMZRSV	39
6.7.2.24 OPC_PROP_ALMZSV	39
6.7.2.25 OPC_PROP_AVI	39
6.7.2.26 OPC_PROP_BGC	40
6.7.2.27 OPC_PROP_BLINK	40
6.7.2.28 OPC_PROP_BMP	40
6.7.2.29 OPC_PROP_CDT	40
6.7.2.30 OPC_PROP_CLOSE	40
6.7.2.31 OPC_PROP_DESC	41
6.7.2.32 OPC_PROP_DTYP	41
6.7.2.33 OPC_PROP_FFST	41
6.7.2.34 OPC_PROP_FGC	41
6.7.2.35 OPC_PROP_HIEU	42
6.7.2.36 OPC_PROP_HIRANGE	42
6.7.2.37 OPC_PROP_HTML	42
6.7.2.38 OPC_PROP_INOUT	42
6.7.2.39 OPC_PROP_INPUT	43
6.7.2.40 OPC_PROP_LOEU	43
6.7.2.41 OPC_PROP_LORANGE	43
6.7.2.42 OPC_PROP_OPEN	43
6.7.2.43 OPC_PROP_OUTPUT	44
6.7.2.44 OPC_PROP_PINI	44
6.7.2.45 OPC_PROP_PLNAME	44
6.7.2.46 OPC_PROP_PREC	44
6.7.2.47 OPC_PROP_QUALITY	45
6.7.2.48 OPC_PROP_RECTYPE	45
6.7.2.49 OPC_PROP_RIGHTS	45
6.7.2.50 OPC_PROP_SCANRATE	45
6.7.2.51 OPC_PROP_SERVER	45
6.7.2.52 OPC_PROP SND	46
6.7.2.53 OPC_PROP_TIME	46
6.7.2.54 OPC_PROP_TIMEZONE	46
6.7.2.55 OPC_PROP_TSE	46
6.7.2.56 OPC_PROP_UNIT	46
6.7.2.57 OPC_PROP_VALUE	47
6.7.2.58 OPC_PROP_ZRST	47
6.8 String functions and classes	48
6.8.1 Detailed Description	48
6.8.2 Typedef Documentation	48

6.8.2.1 stringcase	48
6.8.2.2 wstringcase	49
6.8.3 Function Documentation	49
6.8.3.1 compare() [1/2]	49
6.8.3.2 compare() [2/2]	50
6.8.3.3 eq() [1/2]	50
6.8.3.4 eq() [2/2]	51
6.8.3.5 lt() [1/2]	51
6.8.3.6 lt() [2/2]	51
6.8.3.7 ne() [1/2]	52
6.8.3.8 ne() [2/2]	52
6.8.3.9 split_string()	52
6.8.3.10 strncasecmp()	53
6.8.3.11 trim_space() [1/2]	54
6.8.3.12 trim_space() [2/2]	54
6.8.3.13 wcsncasewcmp()	55
6.9 TwinCAT read/write scanning	56
6.9.1 Detailed Description	56
6.10 EPICS utility functions and classes	57
6.10.1 Detailed Description	57
6.10.2 Typedef Documentation	57
6.10.2.1 filename_set	58
6.10.2.2 macro_list	58
6.10.2.3 macro_stack	58
6.10.3 Enumeration Type Documentation	58
6.10.3.1 case_type	58
6.10.3.2 device_support_type	59
6.10.3.3 io_filestat	59
6.10.3.4 listing_type	59
6.10.3.5 macrofile_type	60
6.10.3.6 tc_epics_conv	60
6.11 EPICS constants	61
6.11.1 Detailed Description	62
6.11.2 Variable Documentation	62
6.11.2.1 EPICS_DB_ALLOWED	62
6.11.2.2 EPICS_DB_COSV	62
6.11.2.3 EPICS_DB_DESC	62
6.11.2.4 EPICS_DB_DRVH	63
6.11.2.5 EPICS_DB_DRVL	63
6.11.2.6 EPICS_DB_DTYP	63
6.11.2.7 EPICS_DB_EGU	63
6.11.2.8 EPICS_DB_FORBIDDEN	64

6.11.2.9 EPICS_DB_HHSV	64
6.11.2.10 EPICS_DB_HIGH	64
6.11.2.11 EPICS_DB_HIHI	64
6.11.2.12 EPICS_DB_HOPR	65
6.11.2.13 EPICS_DB_HSV	65
6.11.2.14 EPICS_DB_HYST	65
6.11.2.15 EPICS_DB_INP	65
6.11.2.16 EPICS_DB_LLSV	66
6.11.2.17 EPICS_DB_LOLO	66
6.11.2.18 EPICS_DB_LOPR	66
6.11.2.19 EPICS_DB_LOW	66
6.11.2.20 EPICS_DB_LSV	67
6.11.2.21 EPICS_DB_MAJOR	67
6.11.2.22 EPICS_DB_MINOR	67
6.11.2.23 EPICS_DB_NOALARM	67
6.11.2.24 EPICS_DB_NUMVAL	68
6.11.2.25 EPICS_DB_ONAM	68
6.11.2.26 EPICS_DB_OSV	68
6.11.2.27 EPICS_DB_OUT	68
6.11.2.28 EPICS_DB_PINI	69
6.11.2.29 EPICS_DB_PREC	69
6.11.2.30 EPICS_DB_SCAN	69
6.11.2.31 EPICS_DB_TSE	69
6.11.2.32 EPICS_DB_UNSV	70
6.11.2.33 EPICS_DB_ZNAM	70
6.11.2.34 EPICS_DB_ZRST	70
6.11.2.35 EPICS_DB_ZRSV	70
6.11.2.36 EPICS_DB_ZRVL	71
6.11.2.37 EPICS_DB_ZSV	71
6.11.2.38 LIGODAQ_DATATYPE_DEFAULT	71
6.11.2.39 LIGODAQ_DATATYPE_FLOAT	71
6.11.2.40 LIGODAQ_DATATYPE_INT32	72
6.11.2.41 LIGODAQ_DATATYPE_NAME	72
6.11.2.42 LIGODAQ_INI_HEADER	72
6.11.2.43 LIGODAQ_UNIT_DEFAULT	72
6.11.2.44 LIGODAQ_UNIT_NAME	73
6.11.2.45 LIGODAQ_UNIT_NONE	73
6.11.2.46 MAX_EPICS_CHANNEL	73
6.11.2.47 MAX_EPICS_DESC	73
6.11.2.48 MAX_EPICS_STRING	73
6.11.2.49 MAX_EPICS_UNIT	73

7 Namespace Documentation	75
7.1 DevTc Namespace Reference	75
7.1.1 Detailed Description	76
7.1.2 Function Documentation	76
7.1.2.1 tc_regex()	77
7.1.2.2 tcAlias()	77
7.1.2.3 tcInfoPrefix()	78
7.1.2.4 tcList()	78
7.1.2.5 tcLoadRecords()	79
7.1.2.6 tcMacro()	79
7.1.2.7 tcPrintVal()	80
7.1.2.8 tcPrintVals()	81
7.1.2.9 tcSetScanRate()	82
7.2 EpicsTpy Namespace Reference	83
7.2.1 Detailed Description	85
7.3 InfoPlc Namespace Reference	85
7.3.1 Detailed Description	85
7.4 ParseTpy Namespace Reference	85
7.4.1 Detailed Description	88
7.4.2 Function Documentation	88
7.4.2.1 compareNamesWoNamespace()	88
7.4.2.2 get_decoration()	89
7.4.2.3 get_pointer()	89
7.5 ParseUtil Namespace Reference	89
7.5.1 Detailed Description	91
7.6 plc Namespace Reference	91
7.6.1 Detailed Description	93
7.6.2 Typedef Documentation	93
7.6.2.1 BasePLCList	93
7.6.2.2 BasePLCPtr	93
7.6.2.3 BaseRecordList	93
7.6.2.4 BaseRecordPtr	94
7.6.2.5 InterfacePtr	94
7.6.3 Enumeration Type Documentation	94
7.6.3.1 access_rights_enum	94
7.6.3.2 data_type_enum	94
7.6.4 Function Documentation	95
7.6.4.1 reset_and_read() [1/3]	95
7.6.4.2 reset_and_read() [2/3]	95
7.6.4.3 reset_and_read() [3/3]	96
7.6.4.4 ScannerProc()	96
7.6.4.5 scannerThread()	97

7.6.4.6 write_and_test() [1/3]	98
7.6.4.7 write_and_test() [2/3]	99
7.6.4.8 write_and_test() [3/3]	99
7.7 TcComms Namespace Reference	99
7.7.1 Detailed Description	100
7.7.2 Function Documentation	101
7.7.2.1 ADSCallback()	101
7.7.2.2 errorPrintf()	101
7.7.2.3 RouterCall()	102
8 Class Documentation	103
8.1 ParseTpy::ads_routing_info Class Reference	103
8.1.1 Detailed Description	104
8.1.2 Member Function Documentation	104
8.1.2.1 get() [1/2]	104
8.1.2.2 get() [2/2]	105
8.1.2.3 set()	105
8.2 TcComms::AmsRouterNotification Class Reference	106
8.2.1 Detailed Description	107
8.2.2 Friends And Related Function Documentation	107
8.2.2.1 RouterCall	107
8.3 std::atomic< string > Class Template Reference	107
8.3.1 Detailed Description	108
8.4 std::atomic< wstring > Class Template Reference	109
8.4.1 Detailed Description	110
8.5 std::atomic_string< stringT > Class Template Reference	110
8.5.1 Detailed Description	111
8.5.2 Member Function Documentation	112
8.5.2.1 operator=()	112
8.6 ParseTpy::base_record Class Reference	112
8.6.1 Detailed Description	113
8.6.2 Constructor & Destructor Documentation	114
8.6.2.1 base_record() [1/4]	114
8.6.2.2 base_record() [2/4]	114
8.6.2.3 base_record() [3/4]	114
8.6.2.4 base_record() [4/4]	115
8.7 plc::BasePLC Class Reference	115
8.7.1 Detailed Description	118
8.7.2 Member Function Documentation	118
8.7.2.1 add() [1/2]	118
8.7.2.2 add() [2/2]	119
8.7.2.3 erase()	120

8.7.2.4 find()	120
8.7.2.5 for_each()	121
8.7.2.6 get_next()	121
8.7.2.7 get_timestamp_unix()	122
8.7.2.8 plc_data_set_valid()	123
8.7.2.9 printRecord()	124
8.7.2.10 reserve()	124
8.7.2.11 start()	124
8.7.2.12 user_data_set_valid()	125
8.7.3 Member Data Documentation	125
8.7.3.1 records	125
8.8 plc::BaseRecord Class Reference	126
8.8.1 Detailed Description	128
8.8.2 Constructor & Destructor Documentation	129
8.8.2.1 BaseRecord() [1/2]	129
8.8.2.2 BaseRecord() [2/2]	129
8.8.3 Member Function Documentation	129
8.8.3.1 PlcGetValid()	129
8.8.3.2 PlcPush()	130
8.8.3.3 PlcRead() [1/3]	131
8.8.3.4 PlcRead() [2/3]	132
8.8.3.5 PlcRead() [3/3]	132
8.8.3.6 PlcReadBinary()	133
8.8.3.7 PlcSetValid()	134
8.8.3.8 PlcWrite() [1/3]	135
8.8.3.9 PlcWrite() [2/3]	136
8.8.3.10 PlcWrite() [3/3]	136
8.8.3.11 PlcWriteBinary()	137
8.8.3.12 UserGetValid()	138
8.8.3.13 UserPush()	138
8.8.3.14 UserRead() [1/3]	140
8.8.3.15 UserRead() [2/3]	141
8.8.3.16 UserRead() [3/3]	142
8.8.3.17 UserReadBinary()	143
8.8.3.18 UserSetValid()	143
8.8.3.19 UserWrite() [1/3]	144
8.8.3.20 UserWrite() [2/3]	145
8.8.3.21 UserWrite() [3/3]	146
8.8.3.22 UserWriteBinary()	146
8.9 ParseUtil::bit_location Class Reference	147
8.9.1 Detailed Description	148
8.10 std::case_char_traits Struct Reference	148

8.10.1 Detailed Description	149
8.11 std::case_wchar_traits Struct Reference	150
8.11.1 Detailed Description	150
8.12 ParseTpy::compiler_info Class Reference	151
8.12.1 Detailed Description	152
8.13 TcComms::DataPar Struct Reference	152
8.13.1 Detailed Description	153
8.14 plc::DataValue Class Reference	153
8.14.1 Detailed Description	156
8.14.2 Constructor & Destructor Documentation	156
8.14.2.1 DataValue()	156
8.14.3 Member Function Documentation	157
8.14.3.1 GetValid()	157
8.14.3.2 Init()	157
8.14.3.3 PlcGetValid()	158
8.14.3.4 PlcRead() [1/4]	159
8.14.3.5 PlcRead() [2/4]	160
8.14.3.6 PlcRead() [3/4]	160
8.14.3.7 PlcRead() [4/4]	161
8.14.3.8 PlcReadBinary()	161
8.14.3.9 PlcSetValid()	162
8.14.3.10 PlcWrite() [1/4]	163
8.14.3.11 PlcWrite() [2/4]	164
8.14.3.12 PlcWrite() [3/4]	165
8.14.3.13 PlcWrite() [4/4]	166
8.14.3.14 PlcWriteBinary()	166
8.14.3.15 Read() [1/5]	167
8.14.3.16 Read() [2/5]	168
8.14.3.17 Read() [3/5]	168
8.14.3.18 Read() [4/5]	169
8.14.3.19 Read() [5/5]	170
8.14.3.20 ReadBinary()	170
8.14.3.21 SetValid()	171
8.14.3.22 UserGetValid()	172
8.14.3.23 UserRead() [1/4]	173
8.14.3.24 UserRead() [2/4]	173
8.14.3.25 UserRead() [3/4]	174
8.14.3.26 UserRead() [4/4]	174
8.14.3.27 UserReadBinary()	175
8.14.3.28 UserSetValid()	176
8.14.3.29 UserWrite() [1/4]	177
8.14.3.30 UserWrite() [2/4]	177

8.14.3.31 UserWrite() [3/4]	178
8.14.3.32 UserWrite() [4/4]	178
8.14.3.33 UserWriteBinary()	179
8.14.3.34 Write() [1/5]	180
8.14.3.35 Write() [2/5]	181
8.14.3.36 Write() [3/5]	182
8.14.3.37 Write() [4/5]	183
8.14.3.38 Write() [5/5]	183
8.14.3.39 WriteBinary()	185
8.14.4 Member Data Documentation	186
8.14.4.1 mysize	186
8.15 plc::DataValueTraits< T > Struct Template Reference	186
8.15.1 Detailed Description	188
8.15.2 Member Data Documentation	188
8.15.2.1 data_enum	188
8.16 plc::DataValueTypeDef Struct Reference	188
8.16.1 Detailed Description	190
8.17 DevTc::devTcDefIn< RecType > Struct Template Reference	190
8.17.1 Detailed Description	192
8.18 DevTc::devTcDefIo< RecType > Struct Template Reference	192
8.18.1 Detailed Description	193
8.19 DevTc::devTcDefOut< RecType > Struct Template Reference	194
8.19.1 Detailed Description	195
8.20 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference	195
8.20.1 Detailed Description	196
8.21 EpicsTpy::epics_conversion Class Reference	196
8.21.1 Detailed Description	198
8.21.2 Constructor & Destructor Documentation	198
8.21.2.1 epics_conversion() [1/3]	198
8.21.2.2 epics_conversion() [2/3]	198
8.21.2.3 epics_conversion() [3/3]	199
8.21.3 Member Function Documentation	199
8.21.3.1 getopt()	199
8.21.3.2 to_epics()	201
8.22 EpicsTpy::epics_db_processing Class Reference	202
8.22.1 Detailed Description	204
8.22.2 Constructor & Destructor Documentation	204
8.22.2.1 epics_db_processing()	204
8.22.3 Member Function Documentation	205
8.22.3.1 getopt()	205
8.22.3.2 mygetopt()	206
8.22.3.3 operator()()	207

8.22.3.4 process_field_alarm()	209
8.22.3.5 process_field_numeric() [1/3]	210
8.22.3.6 process_field_numeric() [2/3]	211
8.22.3.7 process_field_numeric() [3/3]	212
8.22.3.8 process_field_string()	213
8.23 EpicsTpy::epics_list_processing Class Reference	214
8.23.1 Detailed Description	215
8.23.2 Constructor & Destructor Documentation	215
8.23.2.1 epics_list_processing() [1/2]	215
8.23.2.2 epics_list_processing() [2/2]	216
8.23.3 Member Function Documentation	216
8.23.3.1 getopt()	217
8.23.3.2 mygetopt()	217
8.23.3.3 operator()()	218
8.24 EpicsTpy::epics_macrofiles_processing Class Reference	220
8.24.1 Detailed Description	223
8.24.2 Constructor & Destructor Documentation	223
8.24.2.1 epics_macrofiles_processing() [1/2]	223
8.24.2.2 epics_macrofiles_processing() [2/2]	223
8.24.3 Member Function Documentation	224
8.24.3.1 getopt()	224
8.24.3.2 mygetopt()	225
8.24.3.3 operator()()	226
8.24.3.4 to_filename()	227
8.25 DevTc::epics_record_traits< RecType > Struct Template Reference	228
8.25.1 Detailed Description	229
8.26 DevTc::epics_tc_db_processing Class Reference	229
8.26.1 Detailed Description	231
8.26.2 Constructor & Destructor Documentation	231
8.26.2.1 epics_tc_db_processing()	231
8.26.3 Member Function Documentation	231
8.26.3.1 operator()()	232
8.26.3.2 patch_db_recordnames()	232
8.26.3.3 process_list()	232
8.26.3.4 process_lists()	233
8.26.3.5 process_macro()	233
8.26.3.6 process_macros()	233
8.27 DevTc::EpicsInterface Class Reference	234
8.27.1 Detailed Description	237
8.28 std::std::hash< std::stringcase > Struct Template Reference	237
8.28.1 Detailed Description	237
8.28.2 Member Function Documentation	237

8.28.2.1 operator()()	237
8.29 std::std::hash< std::wstringcase > Struct Template Reference	238
8.29.1 Detailed Description	238
8.29.2 Member Function Documentation	238
8.29.2.1 operator()()	238
8.30 InfoPlc::InfoInterface Class Reference	239
8.30.1 Detailed Description	243
8.30.2 Constructor & Destructor Documentation	243
8.30.2.1 InfoInterface() [1/2]	243
8.30.2.2 InfoInterface() [2/2]	243
8.30.3 Member Function Documentation	244
8.30.3.1 get_infodb()	244
8.30.3.2 get_symbol_name()	245
8.30.3.3 printVal()	245
8.30.4 Member Data Documentation	246
8.30.4.1 dbinfo_list	246
8.31 plc::Interface Class Reference	246
8.31.1 Detailed Description	248
8.31.2 Constructor & Destructor Documentation	248
8.31.2.1 Interface()	248
8.31.3 Member Function Documentation	248
8.31.3.1 printVal()	248
8.32 ParseTpy::item_record Class Reference	249
8.32.1 Detailed Description	250
8.33 EpicsTpy::macro_info Struct Reference	250
8.33.1 Detailed Description	251
8.34 EpicsTpy::macro_record Struct Reference	251
8.34.1 Detailed Description	252
8.35 ParseUtil::memory_location Class Reference	252
8.35.1 Detailed Description	253
8.35.2 Constructor & Destructor Documentation	253
8.35.2.1 memory_location() [1/2]	253
8.35.2.2 memory_location() [2/2]	254
8.35.3 Member Function Documentation	254
8.35.3.1 get()	254
8.35.3.2 set()	255
8.35.3.3 set_section()	256
8.36 EpicsTpy::multi_io_support Class Reference	256
8.36.1 Detailed Description	258
8.36.2 Constructor & Destructor Documentation	258
8.36.2.1 multi_io_support()	258
8.36.3 Member Function Documentation	259

8.36.3.1 getopt()	259
8.37 ParseUtil::opc_list Class Reference	260
8.37.1 Detailed Description	261
8.38 ParseUtil::optarg Class Reference	261
8.38.1 Detailed Description	262
8.38.2 Constructor & Destructor Documentation	262
8.38.2.1 optarg()	262
8.38.3 Member Function Documentation	263
8.38.3.1 parse()	263
8.39 ParseTpy::parserinfo_type Class Reference	264
8.39.1 Detailed Description	266
8.39.2 Member Data Documentation	266
8.39.2.1 name_parse	266
8.40 ParseUtil::process_arg Class Reference	267
8.40.1 Detailed Description	268
8.40.2 Constructor & Destructor Documentation	268
8.40.2.1 process_arg()	268
8.40.3 Member Function Documentation	269
8.40.3.1 get_full()	269
8.41 InfoPlc::process_arg_info Class Reference	270
8.41.1 Detailed Description	271
8.41.2 Constructor & Destructor Documentation	271
8.41.2.1 process_arg_info()	271
8.41.3 Member Function Documentation	271
8.41.3.1 get_full()	271
8.42 ParseUtil::process_arg_tc Class Reference	272
8.42.1 Detailed Description	273
8.42.2 Constructor & Destructor Documentation	273
8.42.2.1 process_arg_tc()	274
8.42.3 Member Function Documentation	274
8.42.3.1 get()	274
8.42.3.2 get_full()	275
8.43 ParseTpy::project_record Class Reference	275
8.43.1 Detailed Description	276
8.44 DevTc::register_devsup Class Reference	276
8.44.1 Detailed Description	277
8.45 ParseUtil::replacement_rules Class Reference	278
8.45.1 Detailed Description	279
8.46 plc::scanner_thread_args Struct Reference	279
8.46.1 Detailed Description	280
8.47 EpicsTpy::split_io_support Class Reference	280
8.47.1 Detailed Description	282

8.47.2 Constructor & Destructor Documentation	282
8.47.2.1 split_io_support() [1/2]	282
8.47.2.2 split_io_support() [2/2]	283
8.47.3 Member Function Documentation	283
8.47.3.1 getopt()	283
8.47.3.2 increment()	284
8.47.3.3 operator=()	285
8.48 ParseTpy::symbol_record Class Reference	286
8.48.1 Detailed Description	287
8.49 syminfo_processing Class Reference	287
8.49.1 Detailed Description	287
8.50 plc::System Class Reference	288
8.50.1 Detailed Description	289
8.50.2 Member Function Documentation	289
8.50.2.1 add() [1/2]	289
8.50.2.2 add() [2/2]	289
8.50.2.3 for_each()	290
8.50.2.4 printVal()	290
8.51 ParseUtil::tag_processing Class Reference	291
8.51.1 Detailed Description	292
8.51.2 Constructor & Destructor Documentation	292
8.51.2.1 tag_processing() [1/2]	292
8.51.2.2 tag_processing() [2/2]	293
8.51.3 Member Function Documentation	293
8.51.3.1 getopt()	293
8.52 TcComms::TCatInterface Class Reference	294
8.52.1 Detailed Description	296
8.52.2 Constructor & Destructor Documentation	297
8.52.2.1 TCatInterface()	297
8.52.3 Member Function Documentation	297
8.52.3.1 printVal()	297
8.53 TcComms::TcPLC Class Reference	298
8.53.1 Detailed Description	301
8.53.2 Member Function Documentation	301
8.53.2.1 closePort()	301
8.53.2.2 get_responseBuffer()	302
8.53.2.3 optimizeRequests()	303
8.53.2.4 printRecord()	303
8.53.2.5 set_addr()	304
8.53.2.6 set_ads_state()	304
8.53.2.7 update_scanner()	305
8.53.3 Friends And Related Function Documentation	305

8.53.3.1 ADScallback	305
8.53.4 Member Data Documentation	306
8.53.4.1 cyclesLeft	306
8.54 TcComms::tcProcWrite Class Reference	306
8.54.1 Detailed Description	307
8.54.2 Member Function Documentation	307
8.54.2.1 add()	307
8.54.2.2 read_ptr()	308
8.54.3 Member Data Documentation	309
8.54.3.1 req	309
8.55 DevTc::tcRegisterTolocShell Class Reference	309
8.55.1 Detailed Description	310
8.56 ParseTpy::tpy_file Class Reference	310
8.56.1 Detailed Description	312
8.56.2 Member Function Documentation	312
8.56.2.1 parse_finish()	312
8.56.2.2 process_array()	312
8.56.2.3 process_symbols()	314
8.56.2.4 process_type_tree() [1/3]	315
8.56.2.5 process_type_tree() [2/3]	317
8.56.2.6 process_type_tree() [3/3]	318
8.57 DevTc::epics_record_traits< RecType >::traits_type Struct Reference	320
8.57.1 Detailed Description	320
8.58 ParseTpy::type_map Class Reference	320
8.58.1 Detailed Description	321
8.59 ParseTpy::type_record Class Reference	321
8.59.1 Detailed Description	323
8.60 ParseUtil::variable_name Class Reference	323
8.60.1 Detailed Description	324
9 File Documentation	325
9.1 atomic_string.h File Reference	325
9.1.1 Detailed Description	326
9.2 devTc.cpp File Reference	326
9.2.1 Detailed Description	328
9.3 devTc.h File Reference	328
9.3.1 Detailed Description	330
9.4 drvTc.cpp File Reference	330
9.4.1 Detailed Description	331
9.5 drvTc.h File Reference	332
9.5.1 Detailed Description	332
9.6 EpicsDbGen.cpp File Reference	333

9.6.1 Detailed Description	333
9.6.2 Function Documentation	333
9.6.2.1 main()	333
9.7 infoPlc.cpp File Reference	334
9.7.1 Detailed Description	335
9.8 infoPlc.h File Reference	335
9.8.1 Detailed Description	336
9.9 infoPlcTemplate.h File Reference	336
9.9.1 Detailed Description	337
9.10 iocMain.cpp File Reference	337
9.10.1 Detailed Description	338
9.10.2 Function Documentation	338
9.10.2.1 main()	338
9.11 ParseTpy.cpp File Reference	338
9.11.1 Detailed Description	339
9.12 ParseTpy.h File Reference	339
9.12.1 Detailed Description	340
9.13 ParseTpyConst.h File Reference	341
9.13.1 Detailed Description	343
9.14 ParseTpyInfo.cpp File Reference	343
9.14.1 Detailed Description	343
9.14.2 Function Documentation	344
9.14.2.1 main()	344
9.15 ParseTpyTemplate.h File Reference	344
9.15.1 Detailed Description	345
9.16 ParseUtil.cpp File Reference	345
9.16.1 Detailed Description	345
9.17 ParseUtil.h File Reference	345
9.17.1 Detailed Description	347
9.18 ParseUtilConst.h File Reference	347
9.18.1 Detailed Description	349
9.19 plcBase.cpp File Reference	349
9.19.1 Detailed Description	350
9.20 plcBase.h File Reference	350
9.20.1 Detailed Description	351
9.21 plcBaseTemplate.h File Reference	352
9.21.1 Detailed Description	353
9.22 stdafx.cpp File Reference	353
9.22.1 Detailed Description	353
9.23 stdafx.h File Reference	353
9.23.1 Detailed Description	354
9.24 stringcase.h File Reference	354

9.24.1 Detailed Description	355
9.25 stringcase_hash.h File Reference	355
9.25.1 Detailed Description	356
9.26 tcComms.cpp File Reference	356
9.26.1 Detailed Description	357
9.27 tcComms.h File Reference	357
9.27.1 Detailed Description	358
9.28 TpyToEpics.cpp File Reference	358
9.28.1 Detailed Description	359
9.29 TpyToEpics.h File Reference	359
9.29.1 Detailed Description	360
9.30 TpyToEpicsConst.h File Reference	361
9.30.1 Detailed Description	362
Index	363

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Device support for TwinCAT/ADS	13
Device driver functions	22
Info interface classes and functions	23
TwinCAT tpy file parser	25
XML tpy file constants	28
Parser utility functions and classes	30
OPC property constants	33
String functions and classes	48
TwinCAT read/write scanning	56
EPICS utility functions and classes	57
EPICS constants	61

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

DevTc	Namespace for TCat device support	75
EpicsTpy	Namespace for tpy-db conversion	83
InfoPlc	Namespace for Info communication	85
ParseTpy	Namespace for parsing	85
ParseUtil	Namespace for parsing utilities	89
plc	Namespace for abstract plc functionality	91
TcComms	Namespace for TCat communication	99

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ParseTpy::ads_routing_info	103
ParseTpy::project_record	275
TcComms::AmsRouterNotification	106
std::atomic_string< stringT >	110
std::atomic_string< string >	110
std::atomic< string >	107
std::atomic_string< wstring >	110
std::atomic< wstring >	109
ParseTpy::base_record	112
ParseTpy::item_record	249
ParseTpy::symbol_record	286
ParseTpy::type_record	321
plc::BasePLC	115
TcComms::TcPLC	298
ParseUtil::bit_location	147
ParseTpy::item_record	249
ParseTpy::type_record	321
char_traits	
std::case_char_traits	148
std::case_wchar_traits	150
ParseTpy::compiler_info	151
ParseTpy::project_record	275
TcComms::DataPar	152
plc::DataValueTraits< T >	186
plc::DataValueTypeDef	188
plc::BaseRecord	126
plc::DataValue	153
DevTc::devTcDeflo< RecType >	192
DevTc::devTcDefIn< RecType >	190
DevTc::devTcDefOut< RecType >	194
DevTc::devTcDefWaveformIn< RecType >	195
DevTc::epics_record_traits< RecType >	228
std::std::hash< std::stringcase >	237

std::std::hash< std::wstringcase >	238
plc::Interface	246
DevTc::EpicsInterface	234
InfoPlc::InfoInterface	239
TcComms::TCatInterface	294
EpicsTpy::macro_info	250
EpicsTpy::macro_record	251
ParseUtil::memory_location	252
ParseTpy::symbol_record	286
EpicsTpy::multi_io_support	256
EpicsTpy::epics_macrofiles_processing	220
ParseUtil::opc_list	260
ParseUtil::optarg	261
ParseTpy::parserinfo_type	264
ParseUtil::process_arg	267
InfoPlc::process_arg_info	270
ParseUtil::process_arg_tc	272
DevTc::register_devsup	276
ParseUtil::replacement_rules	278
EpicsTpy::epics_conversion	196
EpicsTpy::epics_db_processing	202
DevTc::epics_tc_db_processing	229
EpicsTpy::epics_list_processing	214
EpicsTpy::epics_macrofiles_processing	220
plc::scanner_thread_args	279
EpicsTpy::split_io_support	280
EpicsTpy::epics_db_processing	202
EpicsTpy::epics_list_processing	214
syminfo_processing	287
plc::System	288
ParseUtil::tag_processing	291
ParseTpy::tpy_file	310
TcComms::tcProcWrite	306
DevTc::tcRegisterTolocShell	309
DevTc::epics_record_traits< RecType >::traits_type	320
type_multipmap	
ParseTpy::type_map	320
ParseUtil::variable_name	323

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ParseTpy::ads_routing_info	ADS routing information	103
TcComms::AmsRouterNotification	AMS Router Notification	106
std::atomic< string >	Atomic<string>	107
std::atomic< wstring >	Atomic<wstring>	109
std::atomic_string< stringT >	Atomic strings	110
ParseTpy::base_record	Base record definition	112
plc::BasePLC	Base PLC	115
plc::BaseRecord	Class for managing a tag/channel	126
ParseUtil::bit_location	Bit location	147
std::case_char_traits	Case insensitive traits	148
std::case_wchar_traits	Case insensitive unicode traits	150
ParseTpy::compiler_info	Compiler information	151
TcComms::DataPar	Memory location struct	152
plc::DataValue	Data value	153
plc::DataValueTraits< T >	Data value traits	186
plc::DataValueTypeDef	Collection of type definitions	188
DevTc::devTcDefIn< RecType >	Device support input record	190
DevTc::devTcDefIo< RecType >	Device support record	192

DevTc::devTcDefOut< RecType >	194
Device support output record	
DevTc::devTcDefWaveformIn< RecType >	195
Device support waveform record	
EpicsTyp::epics_conversion	196
Epics conversion	
EpicsTyp::epics_db_processing	202
Pics database record processing	
EpicsTyp::epics_list_processing	214
List processing	
EpicsTyp::epics_macrofiles_processing	220
Macro file processing	
DevTc::epics_record_traits< RecType >	228
Epics record traits	
DevTc::epics_tc_db_processing	229
EPICS/TCat db processing	
DevTc::EpicsInterface	234
Epics interface class	
std::std::hash< std::stringcase >	237
Hash for case insensitive string	
std::std::hash< std::wstringcase >	238
Hash for case insensitive unicode string	
InfoPlc::InfoInterface	239
Info interface	
plc::Interface	246
Abstract interface	
ParseTyp::item_record	249
Item record	
EpicsTyp::macro_info	250
Macro information	
EpicsTyp::macro_record	251
Macro record	
ParseUtil::memory_location	252
Memory location	
EpicsTyp::multi_io_support	256
Multiple IO support	
ParseUtil::opc_list	260
OPC list	
ParseUtil::optarg	261
Optional arguments	
ParseTyp::parserinfo_type	264
Parser information	
ParseUtil::process_arg	267
Arguments for processing	
InfoPlc::process_arg_info	270
Arguments for processing	
ParseUtil::process_arg_tc	272
Arguments for processing	
ParseTyp::project_record	275
Project information	
DevTc::register_devsup	276
Device support registration	
ParseUtil::replacement_rules	278
Replacement rules	
plc::scanner_thread_args	279
Scanner thread arguments	
EpicsTyp::split_io_support	280
Split IO support	

ParseTpy::symbol_record	
Symbol record	286
syminfo_processing	
Symbol processing	287
plc::System	
System to keep track of PLCs	288
ParseUtil::tag_processing	
Tag processing selection	291
TcComms::TCatInterface	
TCat interface class	294
TcComms::TcPLC	
TwinCAT PLC	298
TcComms::tcProcWrite	
TwinCAT process write requests	306
DevTc::tcRegisterTolocShell	
Register TC commands	309
ParseTpy::tpy_file	
Tpy file parsing	310
DevTc::epics_record_traits< RecType >::traits_type	
Epics record type	320
ParseTpy::type_map	
Type dictionary	320
ParseTpy::type_record	
Type record information	321
ParseUtil::variable_name	
Variable name	323

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

atomic_string.h	325
devTc.cpp	326
devTc.h	328
drvTc.cpp	330
drvTc.h	332
EpicsDbGen.cpp	333
infoPlc.cpp	334
infoPlc.h	335
infoPlcTemplate.h	336
iocMain.cpp	337
ParseTpy.cpp	338
ParseTpy.h	339
ParseTpyConst.h	341
ParseTpyInfo.cpp	343
ParseTpyTemplate.h	344
ParseUtil.cpp	345
ParseUtil.h	345
ParseUtilConst.h	347
plcBase.cpp	349
plcBase.h	350
plcBaseTemplate.h	352
stdafx.cpp	353
stdafx.h	353
stringcase.h	354
stringcase_hash.h	355
tcComms.cpp	356
tcComms.h	357
TpyToEpics.cpp	358
TpyToEpics.h	359
TpyToEpicsConst.h	361

Chapter 6

Module Documentation

6.1 Device support for TwinCAT/ADS

Classes

- class `DevTc::EpicsInterface`
Epics interface class.
- struct `DevTc::epics_record_traits< RecType >::traits_type`
Epics record type.
- struct `DevTc::epics_record_traits< RecType >`
Epics record traits.
- struct `DevTc::devTcDeflo< RecType >`
Device support record.
- struct `DevTc::devTcDefIn< RecType >`
Device support input record.
- struct `DevTc::devTcDefOut< RecType >`
device support output record.
- struct `DevTc::devTcDefWaveformIn< RecType >`
device support waveform record.

Typedefs

- typedef auto `DevTc::register_devsup::link_func(dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
-> bool
Type descriping the link function.
- typedef std::pair< std::regex, link_func & > `DevTc::register_devsup::test_pattern`
pair of pattern and link function
- typedef std::vector< test_pattern > `DevTc::register_devsup::test_pattern_list`
list of pattern/link functions
- typedef epicsFloat64 `DevTc::epics_record_traits< RecType >::value_type`
Value type of (raw) value field.
- typedef `epics_record_traits< RecType >::traits_type DevTc::devTcDeflo< RecType >::rec_type`
Record type: aiRecord, etc.
- typedef `rec_type * DevTc::devTcDeflo< RecType >::rec_type_ptr`
Pointer to record type.

Enumerations

- enum `DevTc::epics_record_enum` {
 `DevTc::aaival` = 0, `DevTc::aaoval`, `DevTc::aival`, `DevTc::aoval`,
 `DevTc::bival`, `DevTc::boval`, `DevTc::eventval`, `DevTc::histogramval`,
 `DevTc::longinval`, `DevTc::longoutval`, `DevTc::mbbival`, `DevTc::mbboval`,
 `DevTc::mbbiDirectval`, `DevTc::mbboDirectval`, `DevTc::stringinval`, `DevTc::stringoutval`,
 `DevTc::waveformval`, `DevTc::airval`, `DevTc::aorval`, `DevTc::birval`,
 `DevTc::borval`, `DevTc::mbbiDirectrval`, `DevTc::mbboDirectrval`, `DevTc::mbbirval`,
 `DevTc::mbborval`, `DevTc::epics_record_enumEnd`, `DevTc::invalidval` = -1 }

Epics record type enum.

Functions

- static void `DevTc::register_devsup::add` (const std::regex &rgx, `link_func` &func)
Register a pattern/link function.
- static bool `DevTc::register_devsup::linkRecord` (const `std::stringcase` &inpout, dbCommon *pEpicsRecord, `plc::BaseRecordPtr` &pRecord)
linkRecord
- `DevTc::register_devsup::register_devsup ()`
Default constructor (adds linkTcRecord entry)
- `DevTc::register_devsup::register_devsup (const register_devsup &)`
Disabled copy constructor.
- `register_devsup & DevTc::register_devsup::operator= (const register_devsup &)`
Disabled assignment operator.
- `DevTc::EpicsInterface::EpicsInterface (plc::BaseRecord &dval)`
Constructor.
- `DevTc::EpicsInterface::~EpicsInterface ()`
Deconstructor.
- void `DevTc::EpicsInterface::set_isPassive` (bool passive)
Set isPassive.
- bool `DevTc::EpicsInterface::get_isCallback` () const
Get isCallback.
- void `DevTc::EpicsInterface::set_isCallback` (bool isCb)
Set isCallback.
- void `DevTc::EpicsInterface::set_pEpicsRecord` (dbCommon *pEpRecord)
Set pEpicsRecord.
- bool `DevTc::EpicsInterface::get_callbackRequestPending` () const
Get callbackRequestPending.
- const CALLBACK & `DevTc::EpicsInterface::callback ()` const
Get pointer to callback structure.
- CALLBACK & `DevTc::EpicsInterface::callback ()`
Get pointer to callback structure.
- const IOSCANPVT & `DevTc::EpicsInterface::ioscan ()` const
Get reference to io scan list pointer.
- IOSCANPVT & `DevTc::EpicsInterface::ioscan ()`
Get reference to io scan list pointer.
- IOSCANPVT `DevTc::EpicsInterface::get_ioscan ()` const
Get pointer to io scan list.
- void `DevTc::EpicsInterface::set_ioscan (const IOSCANPVT ioscan)`
Set pointer to io scan list.

- virtual bool `DevTc::EpicsInterface::push ()` override
Makes a call to the EPICS dbProcess function.
- virtual bool `DevTc::EpicsInterface::pull ()` override
Does nothing.
- static int `DevTc::EpicsInterface::get_callback_queue_size (int pri)`
- static int `DevTc::EpicsInterface::get_callback_queue_used (int pri)`
- static int `DevTc::EpicsInterface::get_callback_queue_free (int pri)`
- void `DevTc::EpicsInterface::ioscan_reset (int bitnum)`
Reset ioscan use flag.
- static const char *const `DevTc::epics_record_traits< RecType >::name ()`
Name of the record.
- static `value_type * DevTc::epics_record_traits< RecType >::val (traits_type *prec)`
Returns the (raw) value of a record.
- static bool `DevTc::epics_record_traits< RecType >::read (traits_type *epicsrec, plc::BaseRecord *baserec)`
Performs the read access on prec.
- static bool `DevTc::epics_record_traits< RecType >::write (plc::BaseRecord *baserec, traits_type *epicsrec)`
Performs the write access on prec.
- `DevTc::devTcDeflo< RecType >::devTcDeflo ()`
Hide constructor.
- static long `DevTc::devTcDeflo< RecType >::get_ioint_info (int cmd, dbCommon *prec, IOSCANPVT *ppvt)`
IO/INT info callback.
- `DevTc::devTcDefIn< RecType >::devTcDefIn ()`
Constructor.
- static long `DevTc::devTcDefIn< RecType >::init_read_record (rec_type_ptr prec)`
init callback for read records
- static long `DevTc::devTcDefIn< RecType >::read (rec_type_ptr precord)`
read callback
- `DevTc::devTcDefOut< RecType >::devTcDefOut ()`
Constructor.
- static long `DevTc::devTcDefOut< RecType >::init_write_record (rec_type_ptr prec)`
init callback for write records
- static long `DevTc::devTcDefOut< RecType >::write (rec_type_ptr precord)`
write callback
- `DevTc::devTcDefWaveformIn< RecType >::devTcDefWaveformIn ()`
Constructor.
- static long `DevTc::devTcDefWaveformIn< RecType >::init_read_waveform_record (rec_type_ptr prec)`
init callback for read records
- static long `DevTc::devTcDefWaveformIn< RecType >::read_waveform (rec_type_ptr precord)`
read callback

Variables

- `test_pattern_list DevTc::register_devsup::tp_list`
list of pattern and links
- static `register_devsup DevTc::register_devsup::the_register_devsup`
the one global instance of the register class
- bool `DevTc::EpicsInterface::isPassive`
- bool `DevTc::EpicsInterface::isCallback`
- dbCommon * `DevTc::EpicsInterface::pEpicsRecord`
Pointer to the EPICS record.

- std::mutex [DevTc::EpicsInterface::ioscanmux](#)
IOSCAN mutex.
- IOSCANPVT [DevTc::EpicsInterface::ioscanpvt](#)
Pointer to IO scan list.
- std::atomic< unsigned int > [DevTc::EpicsInterface::ioscan_inuse](#)
Scan in progress (bit encoded value from priorities)
- CALLBACK [DevTc::EpicsInterface::callbackval](#)
Callback structure.
- double [DevTc::epics_record_traits< RecType >::traits_type::val](#)
Value.
- static const aitEnum [DevTc::epics_record_traits< RecType >::value_ait_type](#) = aitEnumFloat64
- static const aitInt32 [DevTc::epics_record_traits< RecType >::value_count](#) = 0
- static const int [DevTc::epics_record_traits< RecType >::value_conversion](#) = 0
return value for read_io functions 0=default, 2=don't convert
- static const bool [DevTc::epics_record_traits< RecType >::input_record](#) = true
Indicates if this is an input record.
- static const bool [DevTc::epics_record_traits< RecType >::raw_record](#) = false
Indicates if this is a raw record.
- long [DevTc::devTcDeflo< RecType >::number](#)
Number of support functions.
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::report_fn](#)
Report support function.
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::init_fn](#)
Init support function.
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::init_record_fn](#)
Record init support function.
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::get_ioint_info_fn](#)
IO/INT support function.
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::io_fn](#)
Read/write support function.
- DEVSUPFUN [DevTc::devTcDeflo< RecType >::special_linconv_fn](#)
Linear conversion support function.

Friends

- void [DevTc::EpicsInterface::complete_io_scan](#) ([EpicsInterface](#) *, IOSCANPVT, int)

6.1.1 Detailed Description

6.1.2 Enumeration Type Documentation

6.1.2.1 epics_record_enum

```
enum DevTc::epics\_record\_enum
```

Epics record type enum.

This record type enums are used as index the epics traits class

Enumerator

aaival	double input array
aoaval	double output array
aival	double input
aoaval	double output
bival	binary input
boval	binary output
eventval	event
histogramval	histogram
longinval	integer input
longoutval	integer output
mbbival	enum input
mbboval	enum output
mbbiDirectval	enum input direct
mbboDirectval	enum output direct
stringinval	string input
stringoutval	string output
waveformval	waveform
airval	raw double input
aorval	raw double output
birval	raw binary input
borval	raw binary output
mbbiDirectrvl	raw enum input
mbboDirectrvl	raw enum output
mbbirval	raw enum input direct
mbborval	raw enum output direct
epics_record_enumEnd	End of enum (sentinel value)
invalidval	invalid

Definition at line 211 of file devTc.h.

6.1.3 Function Documentation

6.1.3.1 get_callback_queue_free()

```
static int DevTc::EpicsInterface::get_callback_queue_free (
    int pri ) [static]
```

Get the free entries in the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

free entries in the callback ring buffer

6.1.3.2 get_callback_queue_size()

```
static int DevTc::EpicsInterface::get_callback_queue_size (
    int pri ) [static]
```

Get the size of the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

size of the callback ring buffer

6.1.3.3 get_callback_queue_used()

```
static int DevTc::EpicsInterface::get_callback_queue_used (
    int pri ) [static]
```

Get the used entries in the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

used entries in the callback ring buffer

6.1.3.4 linkRecord()

```
bool DevTc::register_devsup::linkRecord (
    const std::string& inpout,
    dbCommon * pEpicsRecord,
    plc::BaseRecordPtr & pRecord ) [static]
```

linkRecord

Go through list and call first link function which matches the pattern Used to link epics records with internal records.

Parameters

<i>inpout</i>	Value of INP/OUT field
<i>pEpicsRecord</i>	Pointer to EPICS record
<i>pRecord</i>	Pointer to a base record (return)

Returns

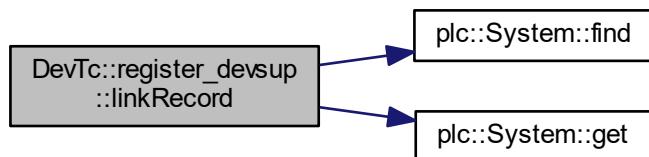
true if one match was found and successfully linked

register_devsup::linkRecord

Definition at line 106 of file devTc.cpp.

References plc::System::find(), and plc::System::get().

Here is the call graph for this function:



6.1.3.5 register_devsup()

`DevTc::register_devsup::register_devsup () [protected]`

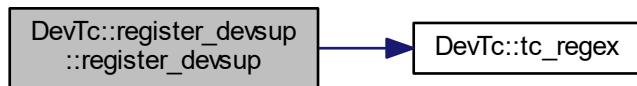
Default constructor (adds linkTcRecord entry)

[register_devsup::register_devsup](#)

Definition at line 98 of file devTc.cpp.

References `DevTc::tc_regex()`.

Here is the call graph for this function:



6.1.4 Variable Documentation

6.1.4.1 isCallback

`bool DevTc::EpicsInterface::isCallback [protected]`

Bool indicating whether callback is needed to call dbProcess true : SCAN = I/O Intr or the record is an out record

Definition at line 194 of file devTc.h.

Referenced by `DevTc::EpicsInterface::get_isCallback()`, `DevTc::EpicsInterface::push()`, and `DevTc::EpicsInterface::set_isCallback()`.

6.1.4.2 isPassive

`bool DevTc::EpicsInterface::isPassive [protected]`

Bool indicating passive scan true : EPICS record SCAN field is set to PASSIVE

Definition at line 191 of file devTc.h.

Referenced by `DevTc::EpicsInterface::push()`, and `DevTc::EpicsInterface::set_isPassive()`.

6.1.4.3 the_register_devsup

```
register_devsup DevTc::register_devsup::the_register_devsup [static], [protected]
```

the one global instance of the register class

[register_devsup::the_register_devsup](#)

Definition at line 76 of file devTc.h.

Referenced by DevTc::register_devsup::add().

6.1.4.4 value_ait_type

```
template<epics_record_enum RecType>
const aitEnum DevTc::epics_record_traits< RecType >::value_ait_type = aitEnumFloat64 [static]
```

Data type of records val/rval field aitEnumInvalid type signals an array. Take type/len from record

Definition at line 286 of file devTc.h.

6.1.4.5 value_count

```
template<epics_record_enum RecType>
const aitInt32 DevTc::epics_record_traits< RecType >::value_count = 0 [static]
```

Array length: 1=scalar value, 0=array - see nelm for length, fixed for strings according to the record

Definition at line 289 of file devTc.h.

6.2 Device driver functions

Namespaces

- [DevTc](#)

Namespace for TCat device support.

6.2.1 Detailed Description

6.3 Info interface classes and functions

Classes

- class `InfoPlc::InfoInterface`
Info interface.
- class `InfoPlc::process_arg_info`
Arguments for processing.

Typedefs

- `typedef bool(InfoInterface::* InfoPlc::info_update_method) ()`
Pointer to info update method.
- `typedef std::tuple< ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list, std::stringcase, bool, update_enum, info_update_method > InfoPlc::info_dbrecord_type`
- `typedef std::vector< info_dbrecord_type > InfoPlc::info_dbrecord_list`
List type of db info tuples.

Enumerations

- enum `InfoPlc::update_enum { InfoPlc::update_enum::forever, InfoPlc::update_enum::once, InfoPlc::update_enum::done }`
Update frequency type.

6.3.1 Detailed Description

6.3.2 Typedef Documentation

6.3.2.1 `info_dbrecord_type`

```
typedef std::tuple<ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list,
std::stringcase, bool, update_enum, info_update_method> InfoPlc::info_dbrecord_type
```

db info tuple with variable name, process type enum, opc list, TwinCAT type string, readonly, update enum, and update method

Definition at line 46 of file infoPlc.h.

6.3.3 Enumeration Type Documentation

6.3.3.1 `update_enum`

```
enum InfoPlc::update_enum [strong]
```

Update frequency type.

Enumerator

forever	Repeat foreveer.
once	once
done	done

Definition at line 27 of file infoPlc.h.

6.4 TwinCAT tpy file parser

Classes

- class [ParseTpy::ads_routing_info](#)
ADS routing information.
- class [ParseTpy::compiler_info](#)
Compiler information.
- class [ParseTpy::project_record](#)
Project information.
- class [ParseTpy::base_record](#)
Base record definition.
- class [ParseTpy::item_record](#)
item record
- class [ParseTpy::type_record](#)
Type record information.
- class [ParseTpy::type_map](#)
Type dictionary.
- class [ParseTpy::symbol_record](#)
Symbol record.
- class [ParseTpy::tpy_file](#)
Tpy file parsing.

Typedefs

- typedef std::pair< int, int > [ParseTpy::dimension](#)
- typedef std::list< [dimension](#) > [ParseTpy::dimensions](#)
- typedef std::map< int, std::stringcase > [ParseTpy::enum_map](#)
- typedef std::pair< int, std::stringcase > [ParseTpy::enum_pair](#)
- typedef std::list< [item_record](#) > [ParseTpy::item_list](#)
- typedef std::multimap< unsigned int, [type_record](#) > [ParseTpy::type_multipmap](#)
- typedef std::list< [symbol_record](#) > [ParseTpy::symbol_list](#)

Enumerations

- enum [ParseTpy::type_enum](#) {
 [ParseTpy::unknown](#), [ParseTpy::simple](#), [ParseTpy::arraytype](#), [ParseTpy::enumtype](#),
 [ParseTpy::structtype](#), [ParseTpy::functionblock](#) }
Type enum.

6.4.1 Detailed Description

6.4.2 Typedef Documentation

6.4.2.1 dimension

```
typedef std::pair<int, int> ParseTyp::dimension
```

This class stores a lbound, elements pair.

Definition at line 221 of file ParseTyp.h.

6.4.2.2 dimensions

```
typedef std::list<dimension> ParseTyp::dimensions
```

This list stores lbound, elements pairs.

Definition at line 225 of file ParseTyp.h.

6.4.2.3 enum_map

```
typedef std::map<int, std::stringcase> ParseTyp::enum_map
```

This map stores a list of enum values.

Definition at line 229 of file ParseTyp.h.

6.4.2.4 enum_pair

```
typedef std::pair<int, std::stringcase> ParseTyp::enum_pair
```

This type stores an enum pair.

Definition at line 233 of file ParseTyp.h.

6.4.2.5 item_list

```
typedef std::list<item_record> ParseTyp::item_list
```

This class stores a list of subitems.

Definition at line 247 of file ParseTyp.h.

6.4.2.6 symbol_list

```
typedef std::list<symbol_record> ParseTpy::symbol_list
```

This is a list of symbol records

Definition at line 354 of file ParseTpy.h.

6.4.2.7 type_multipmap

```
typedef std::multimap<unsigned int, type_record> ParseTpy::type_multipmap
```

This is a multimap to store type records

Definition at line 316 of file ParseTpy.h.

6.4.3 Enumeration Type Documentation

6.4.3.1 type_enum

```
enum ParseTpy::type_enum
```

Type enum.

This structure describes a type record

Enumerator

unknown	Unknown type.
simple	Simple type.
arraytype	Array type.
enumtype	Enumerated type.
structtype	Structure type.
functionblock	Function block.

Definition at line 253 of file ParseTpy.h.

6.5 XML tpy file constants

Variables

- const char *const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"
PLC project info.
- const char *const ParseTpy::xmlProjectInfo = "ProjectInfo"
Project info.
- const char *const ParseTpy::xmlRoutingInfo = "RoutingInfo"
Routing info.
- const char *const ParseTpy::xmlCompilerInfo = "CompilerInfo"
Compiler info.
- const char *const ParseTpy::xmlAdsInfo = "AdsInfo"
ADS info.
- const char *const ParseTpy::xmlDataTypes = "DataTypes"
Data types.
- const char *const ParseTpy::xmlDataType = "DataType"
Data type.
- const char *const ParseTpy::xmlSymbols = "Symbols"
Symbols.
- const char *const ParseTpy::xmlSymbol = "Symbol"
Symbol.
- const char *const ParseTpy::xmlProperties = "Properties"
Properties.
- const char *const ParseTpy::xmlProperty = "Property"
Property.
- const char *const ParseTpy::xmlCompilerVersion = "CompilerVersion"
Compiler version.
- const char *const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"
TwinCAT version.
- const char *const ParseTpy::xmlCpuFamily = "CpuFamily"
CPU family.
- const char *const ParseTpy::xmlNetId = "NetId"
Net ID.
- const char *const ParseTpy::xmlPort = "Port"
Port.
- const char *const ParseTpy::xmlTargetName = "TargetName"
Target name.
- const char *const ParseTpy::xmlName = "Name"
Name.
- const char *const ParseTpy::xmlType = "Type"
Type.
- const char *const ParseTpy::xmlAttrDecoration = "Decoration"
Decoration.
- const char *const ParseTpy::xmlAttrPointer = "Pointer"
Pointer.
- const char *const ParseTpy::xmlIGroup = "IGroup"
I Group.
- const char *const ParseTpy::xmlIOffset = "IOffset"
I Offset.

- const char *const ParseTpy::xmlBitSize = "BitSize"
Bit size.
- const char *const ParseTpy::xmlBitOffs = "BitOffs"
Bit Offset.
- const char *const ParseTpy::xmlArrayInfo = "ArrayInfo"
Array info.
- const char *const ParseTpy::xmlArrayLBound = "LBound"
Lower bound.
- const char *const ParseTpy::xmlArrayElements = "Elements"
Elements.
- const char *const ParseTpy::xmlSubItem = "SubItem"
Sub item.
- const char *const ParseTpy::xmlFbInfo = "FbInfo"
Fb info.
- const char *const ParseTpy::xmlEnumInfo = "EnumInfo"
Enum info.
- const char *const ParseTpy::xmlEnumText = "Text"
Text.
- const char *const ParseTpy::xmlEnumEnum = "Enum"
Enum.
- const char *const ParseTpy::xmlEnumComment = "Comment"
Comment.
- const char *const ParseTpy::xmlValue = "Value"
Value.
- const char *const ParseTpy::xmlDesc = "Desc"
Description.
- const char *const ParseTpy::opcExport = "opc"
OPC.
- const char *const ParseTpy::opcProp = "opc_prop"
OPC property.
- const char *const ParseTpy::opcBracket = "["
OPC bracket.

6.5.1 Detailed Description

6.6 Parser utility functions and classes

Classes

- class [ParseUtil::replacement_rules](#)
Replacement rules.
- class [ParseUtil::optarg](#)
Optional arguments.
- class [ParseUtil::opc_list](#)
OPC list.
- class [ParseUtil::variable_name](#)
Variable name.
- class [ParseUtil::bit_location](#)
Bit location.
- class [ParseUtil::memory_location](#)
Memory location.
- class [ParseUtil::process_arg](#)
Arguments for processing.
- class [ParseUtil::process_arg_tc](#)
Arguments for processing.
- class [ParseUtil::tag_processing](#)
Tag processing selection.

TypeDefs

- `typedef std::map< std::stringcase, std::stringcase > ParseUtil::replacement_table`
- `typedef std::map< int, std::stringcase > ParseUtil::property_map`
- `typedef std::pair< int, std::stringcase > ParseUtil::property_el`

Enumerations

- enum [ParseUtil::opc_enum](#) { `ParseUtil::no_change`, `ParseUtil::publish`, `ParseUtil::silent` }
OPC state enum.
- enum [ParseUtil::process_type_enum](#) {
`ParseUtil::pt_invalid`, `ParseUtil::pt_int`, `ParseUtil::pt_real`, `ParseUtil::pt_bool`,
`ParseUtil::pt_string`, `ParseUtil::pt_enum`, `ParseUtil::pt_binary` }
Process type.
- enum [ParseUtil::process_tag_enum](#) { `ParseUtil::process_all`, `ParseUtil::process_atomic`, `ParseUtil::process_structured` }
Tag preoicessing enum.

6.6.1 Detailed Description

6.6.2 Typedef Documentation

6.6.2.1 property_el

```
typedef std::pair<int, std::stringcase> ParseUtil::property_el
```

This pair stores one element of opc properties.

Definition at line 139 of file ParseUtil.h.

6.6.2.2 property_map

```
typedef std::map<int, std::stringcase> ParseUtil::property_map
```

This map stores a list of opc properties.

Definition at line 135 of file ParseUtil.h.

6.6.2.3 replacement_table

```
typedef std::map<std::stringcase, std::stringcase> ParseUtil::replacement_table
```

Table of replacement rules

Definition at line 21 of file ParseUtil.h.

6.6.3 Enumeration Type Documentation

6.6.3.1 opc_enum

```
enum ParseUtil::opc_enum
```

OPC state enum.

This enum denotes the opc state.

Enumerator

no_change	Do not change inherited behaviour.
publish	Publish.
silent	Do not publish.

Definition at line 123 of file ParseUtil.h.

6.6.3.2 process_tag_enum

```
enum ParseUtil::process_tag_enum
```

Tag preoicessing enum.

Enumerated type to describe the tag processing

Enumerator

process_all	Process all data types.
process_atomic	Process atomic data types.
process_structured	Process structured data type (array, struct, function block)

Definition at line 431 of file ParseUtil.h.

6.6.3.3 process_type_enum

```
enum ParseUtil::process_type_enum
```

Process type.

Enumerated type to describe the process type

Enumerator

pt_invalid	Invalid type.
pt_int	Numeral type.
pt_real	Floating point type.
pt_bool	Logic type.
pt_string	String type.
pt_enum	Enumerated type.
pt_binary	Binary type.

Definition at line 322 of file ParseUtil.h.

6.7 OPC property constants

Variables

- const int ParseUtil::OPC_PROP_CDT = 1
- const int ParseUtil::OPC_PROP_VALUE = 2
- const int ParseUtil::OPC_PROP_QUALITY = 3
- const int ParseUtil::OPC_PROP_TIME = 4
- const int ParseUtil::OPC_PROP_RIGHTS = 5
- const int ParseUtil::OPC_PROP_SCANRATE = 6
- const int ParseUtil::OPC_PROP_UNIT = 100
- const int ParseUtil::OPC_PROP_DESC = 101
- const int ParseUtil::OPC_PROP_HIEU = 102
- const int ParseUtil::OPC_PROP_LOEU = 103
- const int ParseUtil::OPC_PROP_HIRANGE = 104
- const int ParseUtil::OPC_PROP_LORANGE = 105
- const int ParseUtil::OPC_PROP_CLOSE = 106
- const int ParseUtil::OPC_PROP_OPEN = 107
- const int ParseUtil::OPC_PROP_TIMEZONE = 108
- const int ParseUtil::OPC_PROP_FGC = 201
- const int ParseUtil::OPC_PROP_BGC = 202
- const int ParseUtil::OPC_PROP_BLINK = 203
- const int ParseUtil::OPC_PROP_BMP = 204
- const int ParseUtil::OPC_PROP SND = 205
- const int ParseUtil::OPC_PROP_HTML = 206
- const int ParseUtil::OPC_PROP_AVI = 207
- const int ParseUtil::OPC_PROP_ALMSTAT = 300
- const int ParseUtil::OPC_PROP_ALMHELP = 301
- const int ParseUtil::OPC_PROP_ALMAREAS = 302
- const int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303
- const int ParseUtil::OPC_PROP_ALMCCONDITION = 304
- const int ParseUtil::OPC_PROP_ALMLIMIT = 305
- const int ParseUtil::OPC_PROP_ALMDB = 306
- const int ParseUtil::OPC_PROP_ALMH = 307
- const int ParseUtil::OPC_PROP_ALMH = 308
- const int ParseUtil::OPC_PROP_ALML = 309
- const int ParseUtil::OPC_PROP_ALMLL = 310
- const int ParseUtil::OPC_PROP_ALMROC = 311
- const int ParseUtil::OPC_PROP_ALMDEV = 312
- const int ParseUtil::OPC_PROP_PREC = 8500
- const int ParseUtil::OPC_PROP_ZRST = 8510
- const int ParseUtil::OPC_PROP_FFST = 8525
- const int ParseUtil::OPC_PROP_RECTYPE = 8600
- const int ParseUtil::OPC_PROP_INOUT = 8601
- const char *const ParseUtil::OPC_PROP_INPUT = "input"
- const char *const ParseUtil::OPC_PROP_OUTPUT = "output"
- const int ParseUtil::OPC_PROP_TSE = 8602
- const int ParseUtil::OPC_PROP_PINI = 8603
- const int ParseUtil::OPC_PROP_DTYP = 8604
- const int ParseUtil::OPC_PROP_SERVER = 8610
- const int ParseUtil::OPC_PROP_PLNAME = 8611
- const int ParseUtil::OPC_PROP_ALIAS = 8620
- const int ParseUtil::OPC_PROP_ALMOSV = 8700
- const int ParseUtil::OPC_PROP_ALMZSV = 8701

- const int `ParseUtil::OPC_PROP_ALMCOSV` = 8702
- const int `ParseUtil::OPC_PROP_ALMUNSV` = 8703
- const int `ParseUtil::OPC_PROP_ALMZRSV` = 8710
- const int `ParseUtil::OPC_PROP_ALMFFSV` = 8725
- const int `ParseUtil::OPC_PROP_ALMHHSV` = 8727
- const int `ParseUtil::OPC_PROP_ALMHHSV` = 8728
- const int `ParseUtil::OPC_PROP_ALMLSV` = 8729
- const int `ParseUtil::OPC_PROP_ALMLLSV` = 8730

6.7.1 Detailed Description

6.7.2 Variable Documentation

6.7.2.1 OPC_PROP_ALIAS

`const int ParseUtil::OPC_PROP_ALIAS` = 8620

alias for structure item or symbol name

Definition at line 69 of file `ParseUtilConst.h`.

Referenced by `ParseUtil::variable_name::append()`, and `EpicsTpy::epics_db_processing::operator()()`.

6.7.2.2 OPC_PROP_ALMAREAS

`const int ParseUtil::OPC_PROP_ALMAREAS` = 302

area

Definition at line 45 of file `ParseUtilConst.h`.

6.7.2.3 OPC_PROP_ALMCONDITION

`const int ParseUtil::OPC_PROP_ALMCONDITION` = 304

condition

Definition at line 47 of file `ParseUtilConst.h`.

6.7.2.4 OPC_PROP_ALMCOSV

```
const int ParseUtil::OPC_PROP_ALMCOSV = 8702
```

alarm: change of state severity

Definition at line 72 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.5 OPC_PROP_ALMDB

```
const int ParseUtil::OPC_PROP_ALMDB = 306
```

dead band, tolerance

Definition at line 49 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.6 OPC_PROP_ALMDEV

```
const int ParseUtil::OPC_PROP_ALMDEV = 312
```

deviation

Definition at line 55 of file ParseUtilConst.h.

6.7.2.7 OPC_PROP_ALMFFSV

```
const int ParseUtil::OPC_PROP_ALMFFSV = 8725
```

alarm: fifteen state severity

Definition at line 75 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.8 OPC_PROP_ALMH

```
const int ParseUtil::OPC_PROP_ALMH = 308
```

high alarm -> HIGH

Definition at line 51 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.9 OPC_PROP_ALMHELP

```
const int ParseUtil::OPC_PROP_ALMHELP = 301
```

help

Definition at line 44 of file ParseUtilConst.h.

6.7.2.10 OPC_PROP_ALMHH

```
const int ParseUtil::OPC_PROP_ALMHH = 307
```

high high alarm -> HIHI

Definition at line 50 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.11 OPC_PROP_ALMHHSV

```
const int ParseUtil::OPC_PROP_ALMHHSV = 8727
```

alarm: hihi severity

Definition at line 76 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.12 OPC_PROP_ALMHSV

```
const int ParseUtil::OPC_PROP_ALMHSV = 8728
```

alarm: high severity

Definition at line 77 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.13 OPC_PROP_ALML

```
const int ParseUtil::OPC_PROP_ALML = 309
```

low alarm -> LOW

Definition at line 52 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.14 OPC_PROP_ALMLIMIT

```
const int ParseUtil::OPC_PROP_ALMLIMIT = 305
```

limit

Definition at line 48 of file ParseUtilConst.h.

6.7.2.15 OPC_PROP_ALMLL

```
const int ParseUtil::OPC_PROP_ALMLL = 310
```

low low alaam -> LOLO

Definition at line 53 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.16 OPC_PROP_ALMLLSV

```
const int ParseUtil::OPC_PROP_ALMLLSV = 8730
```

alarm: lolo severity

Definition at line 79 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.17 OPC_PROP_ALMLSV

```
const int ParseUtil::OPC_PROP_ALMLSV = 8729
```

alarm: low severity

Definition at line 78 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.18 OPC_PROP_ALMOSV

```
const int ParseUtil::OPC_PROP_ALMOSV = 8700
```

alarm: one severity

Definition at line 70 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.19 OPC_PROP_ALMPRIMARYAREA

```
const int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303
```

primery area

Definition at line 46 of file ParseUtilConst.h.

6.7.2.20 OPC_PROP_ALMROC

```
const int ParseUtil::OPC_PROP_ALMROC = 311
```

rate of change

Definition at line 54 of file ParseUtilConst.h.

6.7.2.21 OPC_PROP_ALMSTAT

```
const int ParseUtil::OPC_PROP_ALMSTAT = 300  
status
```

Definition at line 43 of file ParseUtilConst.h.

6.7.2.22 OPC_PROP_ALMUNSV

```
const int ParseUtil::OPC_PROP_ALMUNSV = 8703  
alarm: unknown state severity
```

Definition at line 73 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.23 OPC_PROP_ALMZRSV

```
const int ParseUtil::OPC_PROP_ALMZRSV = 8710  
alarm: one state severity
```

Definition at line 74 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.24 OPC_PROP_ALMZSV

```
const int ParseUtil::OPC_PROP_ALMZSV = 8701  
alarm: zero severity
```

Definition at line 71 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.25 OPC_PROP_AVI

```
const int ParseUtil::OPC_PROP_AVI = 207  
avi file
```

Definition at line 41 of file ParseUtilConst.h.

6.7.2.26 OPC_PROP_BGC

const int ParseUtil::OPC_PROP_BGC = 202
background color

Definition at line 36 of file ParseUtilConst.h.

6.7.2.27 OPC_PROP_BLINK

const int ParseUtil::OPC_PROP_BLINK = 203
blinking

Definition at line 37 of file ParseUtilConst.h.

6.7.2.28 OPC_PROP_BMP

const int ParseUtil::OPC_PROP_BMP = 204
bmp file

Definition at line 38 of file ParseUtilConst.h.

6.7.2.29 OPC_PROP_CDT

const int ParseUtil::OPC_PROP_CDT = 1
canonocal data type

Definition at line 18 of file ParseUtilConst.h.

6.7.2.30 OPC_PROP_CLOSE

const int ParseUtil::OPC_PROP_CLOSE = 106
label for close state -> ONAM

Definition at line 31 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.7.2.31 OPC_PROP_DESC

```
const int ParseUtil::OPC_PROP_DESC = 101
```

description string -> DESC

Definition at line 26 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.32 OPC_PROP_DTYP

```
const int ParseUtil::OPC_PROP_DTYP = 8604
```

DTYP field: opc or opcRaw

Definition at line 66 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.33 OPC_PROP_FFST

```
const int ParseUtil::OPC_PROP_FFST = 8525
```

... fifteen string

Definition at line 59 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.7.2.34 OPC_PROP_FGC

```
const int ParseUtil::OPC_PROP_FGC = 201
```

foreground color

Definition at line 35 of file ParseUtilConst.h.

6.7.2.35 OPC_PROP_HIEU

const int ParseUtil::OPC_PROP_HIEU = 102

high expectation value -> HOPR

Definition at line 27 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.36 OPC_PROP_HIRANGE

const int ParseUtil::OPC_PROP_HIRANGE = 104

absolute maximum value -> DRVH

Definition at line 29 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.37 OPC_PROP_HTML

const int ParseUtil::OPC_PROP_HTML = 206

html file

Definition at line 40 of file ParseUtilConst.h.

6.7.2.38 OPC_PROP_INOUT

const int ParseUtil::OPC_PROP_INOUT = 8601

input or output

Definition at line 61 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY(), and EpicsTpy::epics_db_processing::operator()().

6.7.2.39 OPC_PROP_INPUT

```
const char* const ParseUtil::OPC_PROP_INPUT = "input"
```

input

Definition at line 62 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.7.2.40 OPC_PROP_LOEU

```
const int ParseUtil::OPC_PROP_LOEU = 103
```

low expectation value -> LOPR

Definition at line 28 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.41 OPC_PROP_LORANGE

```
const int ParseUtil::OPC_PROP_LORANGE = 105
```

absolute minimum value -> DRVL

Definition at line 30 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.42 OPC_PROP_OPEN

```
const int ParseUtil::OPC_PROP_OPEN = 107
```

label for open state -> ZNAM

Definition at line 32 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.7.2.43 OPC_PROP_OUTPUT

```
const char* const ParseUtil::OPC_PROP_OUTPUT = "output"
```

output

Definition at line 63 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.7.2.44 OPC_PROP_PINI

```
const int ParseUtil::OPC_PROP_PINI = 8603
```

initialization

Definition at line 65 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.45 OPC_PROP_PLNAME

```
const int ParseUtil::OPC_PROP_PLNAME = 8611
```

tc name including ads routing info and port

Definition at line 68 of file ParseUtilConst.h.

Referenced by ParseUtil::process_arg_tc::get_full(), EpicsTpy::epics_db_processing::operator()(), and ParseTpy::tpy_file::parse_finish().

6.7.2.46 OPC_PROP_PREC

```
const int ParseUtil::OPC_PROP_PREC = 8500
```

precision

Definition at line 57 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.47 OPC_PROP_QUALITY

```
const int ParseUtil::OPC_PROP_QUALITY = 3
```

data quality flag

Definition at line 20 of file ParseUtilConst.h.

6.7.2.48 OPC_PROP_RECTYPE

```
const int ParseUtil::OPC_PROP_RECTYPE = 8600
```

record type

Definition at line 60 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.49 OPC_PROP_RIGHTS

```
const int ParseUtil::OPC_PROP_RIGHTS = 5
```

access right: 1 read, 2 write, 3 read/write

Definition at line 22 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.7.2.50 OPC_PROP_SCANRATE

```
const int ParseUtil::OPC_PROP_SCANRATE = 6
```

scan rate

Definition at line 23 of file ParseUtilConst.h.

6.7.2.51 OPC_PROP_SERVER

```
const int ParseUtil::OPC_PROP_SERVER = 8610
```

server name

Definition at line 67 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.52 OPC_PROP_SND

```
const int ParseUtil::OPC_PROP_SND = 205  
sound file
```

Definition at line 39 of file ParseUtilConst.h.

6.7.2.53 OPC_PROP_TIME

```
const int ParseUtil::OPC_PROP_TIME = 4  
timestamp
```

Definition at line 21 of file ParseUtilConst.h.

6.7.2.54 OPC_PROP_TIMEZONE

```
const int ParseUtil::OPC_PROP_TIMEZONE = 108  
time zone
```

Definition at line 33 of file ParseUtilConst.h.

6.7.2.55 OPC_PROP_TSE

```
const int ParseUtil::OPC_PROP_TSE = 8602  
time stamp
```

Definition at line 64 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.7.2.56 OPC_PROP_UNIT

```
const int ParseUtil::OPC_PROP_UNIT = 100  
unit string -> EGU
```

Definition at line 25 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.7.2.57 OPC_PROP_VALUE

```
const int ParseUtil::OPC_PROP_VALUE = 2  
value
```

Definition at line 19 of file ParseUtilConst.h.

6.7.2.58 OPC_PROP_ZRST

```
const int ParseUtil::OPC_PROP_ZRST = 8510  
zero string ...
```

Definition at line 58 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.8 String functions and classes

Classes

- struct `std::case_char_traits`
case insensitive traits.
- struct `std::case_wchar_traits`
case insensitive unicode traits.

Typedefs

- `typedef std::basic_string< char, case_char_traits > std::stringcase`
case insensitive string.
- `typedef std::basic_string< wchar_t, case_wchar_traits > std::wstringcase`
case insensitive string.

Functions

- int `std::strncasecmp` (const char *s1, const char *s2, size_t n)
case insensitive compare with maximum length
- int `std::wcscasewcmp` (const wchar_t *s1, const wchar_t *s2, size_t n)
case insensitive unicode compare with maximum length
- static bool `std::case_char_traits::eq` (const char_type &c1, const char_type &c2)
- static bool `std::case_char_traits::ne` (const char_type &c1, const char_type &c2)
- static bool `std::case_char_traits::lt` (const char_type &c1, const char_type &c2)
- static int `std::case_char_traits::compare` (const char_type *s1, const char_type *s2, size_t n)
- static bool `std::case_wchar_traits::eq` (const char_type &c1, const char_type &c2)
- static bool `std::case_wchar_traits::ne` (const char_type &c1, const char_type &c2)
- static bool `std::case_wchar_traits::lt` (const char_type &c1, const char_type &c2)
- static int `std::case_wchar_traits::compare` (const char_type *s1, const char_type *s2, size_t n)
- void `std::trim_space` (`std::stringcase &s`)
- void `std::trim_space` (`std::wstringcase &s`)
- template<class Container , class String , class Predicate >
`void std::split_string` (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)
Splits a strings.

6.8.1 Detailed Description

6.8.2 Typedef Documentation

6.8.2.1 `stringcase`

```
typedef std::basic_string<char, case_char_traits> std::stringcase  

case insensitive string.
```

This string class is not case sensitive.

Definition at line 104 of file stringcase.h.

6.8.2.2 wstringcase

```
typedef std::basic_string<wchar_t, case_wchar_traits> std::wstringcase
```

case insensitive string.

This string class is not case sensitive.

Definition at line 109 of file stringcase.h.

6.8.3 Function Documentation

6.8.3.1 compare() [1/2]

```
static int std::case_char_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static]
```

Compare strings

Parameters

s1	First string
s2	Second string
n	number of characters

Definition at line 64 of file stringcase.h.

References std::strncasecmp().

Here is the call graph for this function:



6.8.3.2 compare() [2/2]

```
static int std::case_wchar_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static]
```

Compare strings

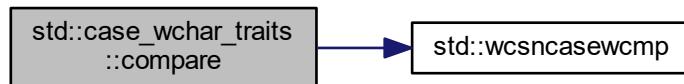
Parameters

<i>s1</i>	First string
<i>s2</i>	Second string
<i>n</i>	number of characters

Definition at line 96 of file stringcase.h.

References std::wcsncasewcmp().

Here is the call graph for this function:



6.8.3.3 eq() [1/2]

```
static bool std::case_char_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 45 of file stringcase.h.

6.8.3.4 eq() [2/2]

```
static bool std::case_wchar_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 77 of file stringcase.h.

6.8.3.5 lt() [1/2]

```
static bool std::case_char_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Lower than character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 57 of file stringcase.h.

6.8.3.6 lt() [2/2]

```
static bool std::case_wchar_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Lower than character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 89 of file stringcase.h.

6.8.3.7 ne() [1/2]

```
static bool std::case_char_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Not equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 51 of file stringcase.h.

6.8.3.8 ne() [2/2]

```
static bool std::case_wchar_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Not equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 83 of file stringcase.h.

6.8.3.9 split_string()

```
template<class Container , class String , class Predicate >
void std::split_string (
    Container & output,
    const String & input,
    const Predicate & pred,
    bool trimEmpty = true )
```

Splits a strings.

Splits a string into its tokens and adds them to a container. The delimiter can be easily specified with a lambda expression. Example: `stringcase arg ("This is a test!"); vector<stringcase> list; split_string (list, arg, [] (char c)->bool { return isspace (c) != 0; }, true);`

Parameters

<i>output</i>	Output container
<i>input</i>	Input string
<i>pred</i>	Function which returns true when character is a separator
<i>trimEmpty</i>	Trims empty strings when true

Definition at line 135 of file stringcase.h.

Referenced by ParseUtil::optarg::parse().

Here is the caller graph for this function:



6.8.3.10 strncasecmp()

```
int std::strncasecmp (
    const char * s1,
    const char * s2,
    size_t n ) [inline]
```

case insensitive compare with maximum length

Case insensitive compare.

Parameters

<i>s1</i>	first string
<i>s2</i>	second string
<i>n</i>	Number of characters

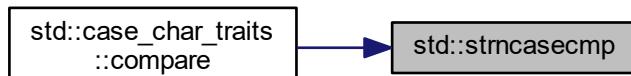
Returns

<0 smaller, 0 equal, >0 greater

Definition at line 23 of file stringcase.h.

Referenced by std::case_char_traits::compare().

Here is the caller graph for this function:



6.8.3.11 trim_space() [1/2]

```
void std::trim_space (
    std::stringcase & s )
```

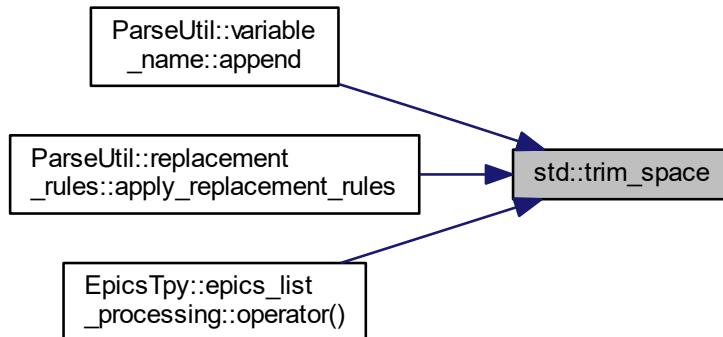
trim space on both ends.

Parameters

s	string to trim
---	----------------

Referenced by ParseUtil::variable_name::append(), ParseUtil::replacement_rules::apply_replacement_rules(), and EpicsTpy::epics_list_processing::operator()().

Here is the caller graph for this function:



6.8.3.12 trim_space() [2/2]

```
void std::trim_space (
    std::wstringcase & s )
```

trim space on both ends.

Parameters

s	string to trim
---	----------------

6.8.3.13 wcsncasewcmp()

```
int std::wcsncasewcmp (
    const wchar_t * s1,
    const wchar_t * s2,
    size_t n ) [inline]
```

case insensitive unicode compare with maximum length

Case insensitive compare for unicode string.

Parameters

<i>s1</i>	first string
<i>s2</i>	second string
<i>n</i>	Number of characters

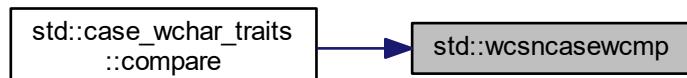
Returns

<0 smaller, 0 equal, >0 greater

Definition at line 34 of file stringcase.h.

Referenced by std::case_wchar_traits::compare().

Here is the caller graph for this function:



6.9 TwinCAT read/write scanning

Classes

- struct **TcComms::DataPar**
Memory location struct.
- class **TcComms::TCatInterface**
TCat interface class.
- class **TcComms::tcProcWrite**
TwinCAT process write requests.
- class **TcComms::TcPLC**
TwinCAT PLC.
- class **TcComms::AmsRouterNotification**
AMS Router Notification.

Variables

- const int **TcComms::MAX_REQ_SIZE** = 250000
maximum allowed request size (bytes)
- const int **TcComms::MAX_SINGLE_GAP_SIZE** = 50
maximum allowed size (bytes) of a memory gap within continuous request
- const double **TcComms::MAX_REL_GAP** = 0.25
(maximum allowed total gap size) / (current request size)
- const int **TcComms::MIN_REL_GAP_SIZE** = 100
minimum allowed relative gap size (bytes)
- const int **TcComms::default_scanrate** = 100
default PLC TwinCAT scan rate (100ms)
- const int **TcComms::minimum_scanrate** = 5
minimum PLC TwinCAT scan rate (5ms)
- const int **TcComms::maximum_scanrate** = 10000
maximum PLC TwinCAT scan rate (10s)
- const int **TcComms::default_multiple** = 10
default multiple for PLC EPICS scan rate (10)
- const int **TcComms::minimum_multiple** = 1
minimum multiple for PLC EPICS scan rate (1)
- const int **TcComms::maximum_multiple** = 200
maximum multiple for PLC EPICS scan rate (200)

6.9.1 Detailed Description

6.10 EPICS utility functions and classes

Classes

- class [EpicsTpy::epics_conversion](#)
Epics conversion.
- class [EpicsTpy::split_io_support](#)
Split IO support.
- class [EpicsTpy::multi_io_support](#)
Multiple IO support.
- class [EpicsTpy::epics_list_processing](#)
List processing.
- struct [EpicsTpy::macro_info](#)
Macro information.
- struct [EpicsTpy::macro_record](#)
Macro record.
- class [EpicsTpy::epics_macros_processing](#)
Macro file processing.
- class [EpicsTpy::epics_db_processing](#)
epics database record processing

Typedefs

- typedef std::vector< [macro_info](#) > [EpicsTpy::macro_list](#)
- typedef std::stack< [macro_record](#) > [EpicsTpy::macro_stack](#)
- typedef std::unordered_set< std::stringcase > [EpicsTpy::filename_set](#)

Enumerations

- enum [EpicsTpy::tc_epics_conv](#) { [EpicsTpy::no_conversion](#), [EpicsTpy::no_dot](#), [EpicsTpy::ligo_std](#), [EpicsTpy::ligo_vac](#) }
Conversion rules for TC/EPICS.
- enum [EpicsTpy::case_type](#) { [EpicsTpy::preserve_case](#), [EpicsTpy::upper_case](#), [EpicsTpy::lower_case](#) }
Case conversion rule enum.
- enum [EpicsTpy::io_filestat](#) { [EpicsTpy::io_filestat::closed](#), **read**, **write** }
enum for file io
- enum [EpicsTpy::listing_type](#) { [EpicsTpy::listing_standard](#), [EpicsTpy::listing_autoburt](#), [EpicsTpy::listing_daqini](#) }
Listing type enum.
- enum [EpicsTpy::macrofile_type](#) { [EpicsTpy::macrofile_type::all](#), [EpicsTpy::macrofile_type::fields](#), [EpicsTpy::macrofile_type::error](#) }
- enum [EpicsTpy::device_support_type](#) { [EpicsTpy::device_support_opc_name](#), [EpicsTpy::device_support_tc_name](#) }
Device support enum.

6.10.1 Detailed Description

6.10.2 Typedef Documentation

6.10.2.1 filename_set

```
typedef std::unordered_set<std::stringcase> EpicsTpy::filename_set
```

A set of filenames

Definition at line 559 of file TpyToEpics.h.

6.10.2.2 macro_list

```
typedef std::vector<macro_info> EpicsTpy::macro_list
```

A list of fields

Definition at line 532 of file TpyToEpics.h.

6.10.2.3 macro_stack

```
typedef std::stack<macro_record> EpicsTpy::macro_stack
```

A stack of records/structs

Definition at line 555 of file TpyToEpics.h.

6.10.3 Enumeration Type Documentation

6.10.3.1 case_type

```
enum EpicsTpy::case_type
```

Case conversion rule enum.

This enum describes the case conversion rule

Enumerator

preserve_case	Preserve the case.
upper_case	Convert to upper case.
lower_case	Convert to lower case.

Definition at line 46 of file TpyToEpics.h.

6.10.3.2 device_support_type

```
enum EpicsTpy::device_support_type
```

Device support enum.

This enum describes the type of listing to produce

Enumerator

device_support_opc_name	Use opc names in the INPUT/OUTPUT epics fields.
device_support_tc_name	Use TwinCAT names in the INPUT/OUTPUT epics fields.

Definition at line 684 of file TpyToEpics.h.

6.10.3.3 io_filestat

```
enum EpicsTpy::io_filestat [strong]
```

enum for file io

Enumerator

closed	file is closed
--------	----------------

Definition at line 300 of file TpyToEpics.h.

6.10.3.4 listing_type

```
enum EpicsTpy::listing_type
```

Listing type enum.

This enum describes the type of listing to produce

Enumerator

listing_standard	Standard listing using TwinCAT/OPC names.
listing_autoburt	Autoburt listing.
listing_daqini	LIGO DAQ ini listing.

Definition at line 415 of file TpyToEpics.h.

6.10.3.5 macrofile_type

```
enum EpicsTpy::macrofile_type [strong]
```

This enum describes the type of macros to produce

Enumerator

all	Include all fields and error messages.
fields	Include all fields.
errors	Include error messages.

Definition at line 504 of file TpyToEpics.h.

6.10.3.6 tc_epics_conv

```
enum EpicsTpy::tc_epics_conv
```

Conversion rules for TC/EPICS.

This enum describes the TwinCAT/opc to EPICS conversion rule

Enumerator

no_conversion	No conversion.
no_dot	Convert '.' to '_'.
ligo_std	LIGO standard conversion: Eliminate leading '.', Replace sceond '.' with ':', Replace third '.' with '-', Replace all other '.' with '-'.
ligo_vac	LIGO standard conversion for vacuum channels: Eliminate leading '.', Replace sceond '.' with '-', Replace third '.' with '--'.

Definition at line 24 of file TpyToEpics.h.

6.11 EPICS constants

Variables

- const int `EpicsTpy::MAX_EPICS_CHANNEL` = 54
- const int `EpicsTpy::MAX_EPICS_DESC` = 40
- const int `EpicsTpy::MAX_EPICS_STRING` = 40
- const int `EpicsTpy::MAX_EPICS_UNIT` = 15
- const char *const `EpicsTpy::EPICS_DB_EGU` = "EGU"
- const char *const `EpicsTpy::EPICS_DB_DESC` = "DESC"
- const char *const `EpicsTpy::EPICS_DB_HOPR` = "HOPR"
- const char *const `EpicsTpy::EPICS_DB_LOPR` = "LOPR"
- const char *const `EpicsTpy::EPICS_DB_DRVH` = "DRVH"
- const char *const `EpicsTpy::EPICS_DB_DRVL` = "DRVL"
- const char *const `EpicsTpy::EPICS_DB_ONAM` = "ONAM"
- const char *const `EpicsTpy::EPICS_DB_ZNAM` = "ZNAM"
- const char *const `EpicsTpy::EPICS_DB_PREC` = "PREC"
- const char *const `EpicsTpy::EPICS_DB_ZRST` [16]
- const char *const `EpicsTpy::EPICS_DB_ZRVL` [16]
- const char *const `EpicsTpy::EPICS_DB_SCAN` = "SCAN"
- const char *const `EpicsTpy::EPICS_DB_INP` = "INP"
- const char *const `EpicsTpy::EPICS_DB_OUT` = "OUT"
- const char *const `EpicsTpy::EPICS_DB_TSE` = "TSE"
- const char *const `EpicsTpy::EPICS_DB_PINI` = "PINI"
- const char *const `EpicsTpy::EPICS_DB_DTYP` = "DTYP"
- const char *const `EpicsTpy::EPICS_DB_OSV` = "OSV"
- const char *const `EpicsTpy::EPICS_DB_ZSV` = "ZSV"
- const char *const `EpicsTpy::EPICS_DB_COSV` = "COSV"
- const char *const `EpicsTpy::EPICS_DB_HIHI` = "HIHI"
- const char *const `EpicsTpy::EPICS_DB_HIGH` = "HIGH"
- const char *const `EpicsTpy::EPICS_DB_LOW` = "LOW"
- const char *const `EpicsTpy::EPICS_DB_LOLO` = "LOLO"
- const char *const `EpicsTpy::EPICS_DB_HYST` = "HYST"
- const char *const `EpicsTpy::EPICS_DB_HHSV` = "HHSV"
- const char *const `EpicsTpy::EPICS_DB_HSV` = "HSV"
- const char *const `EpicsTpy::EPICS_DB_LSV` = "LSV"
- const char *const `EpicsTpy::EPICS_DB_LLSV` = "LLSV"
- const char *const `EpicsTpy::EPICS_DB_NOALARM` = "NO_ALARM"
- const char *const `EpicsTpy::EPICS_DB_MINOR` = "MINOR"
- const char *const `EpicsTpy::EPICS_DB_MAJOR` = "MAJOR"
- const char *const `EpicsTpy::EPICS_DB_UNSV` = "UNSV"
- const char *const `EpicsTpy::EPICS_DB_ZRSV` [16]
- const char *const `EpicsTpy::EPICS_DB_FORBIDDEN` []
- const char *const `EpicsTpy::EPICS_DB_ALLOWED` []
- const char *const `EpicsTpy::EPICS_DB_NUMVAL` []
- const char *const `EpicsTpy::LIGODAQ_DATATYPE_NAME` = "datatype"
- const int `EpicsTpy::LIGODAQ_DATATYPE_FLOAT` = 4
- const int `EpicsTpy::LIGODAQ_DATATYPE_INT32` = 2
- const int `EpicsTpy::LIGODAQ_DATATYPE_DEFAULT` = `LIGODAQ_DATATYPE_FLOAT`
- const char *const `EpicsTpy::LIGODAQ_UNIT_NAME` = "units"
- const char *const `EpicsTpy::LIGODAQ_UNIT_NONE` = "none"
- const char *const `EpicsTpy::LIGODAQ_UNIT_DEFAULT` = `LIGODAQ_UNIT_NONE`
- const char *const `EpicsTpy::LIGODAQ_INI_HEADER`

6.11.1 Detailed Description

6.11.2 Variable Documentation

6.11.2.1 EPICS_DB_ALLOWED

```
const char* const EpicsTpy::EPICS_DB_ALLOWED[ ]
```

Initial value:

```
= {"LINR", "EGUF", "EGUL", "AOFF", "ASLO", "ESLO", "SMOO", "HIHI", "LOLO", "HIGH", "LOW", "HHSV", "LLSV",
     "HSV", "LSV", "HYST", "ZSV", "OSV", "COSV", "IVOA", "IVOV", "UNSV", "ZRSV",
     "ONSV", "TWSV", "THSV",
     "FRSV", "FVSV", "SXSV", "SVSV", "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV",
     "FTSV", "FFSV", NULL}
```

Names of allowed EPICS record fields

Definition at line 75 of file TpyToEpicsConst.h.

6.11.2.2 EPICS_DB_COSV

```
const char* const EpicsTpy::EPICS_DB_COSV = "COSV"
```

change severity

Definition at line 47 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.3 EPICS_DB_DESC

```
const char* const EpicsTpy::EPICS_DB_DESC = "DESC"
```

description string

Definition at line 24 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.4 EPICS_DB_DRVH

```
const char* const EpicsTpy::EPICS_DB_DRVH = "DRVH"
```

drive high string

Definition at line 27 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.5 EPICS_DB_DRVL

```
const char* const EpicsTpy::EPICS_DB_DRVL = "DRVL"
```

drive low string

Definition at line 28 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.6 EPICS_DB_DTYP

```
const char* const EpicsTpy::EPICS_DB_DTYP = "DTYP"
```

data type

Definition at line 43 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.7 EPICS_DB_EGU

```
const char* const EpicsTpy::EPICS_DB_EGU = "EGU"
```

unit string

Definition at line 23 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.8 EPICS_DB_FORBIDDEN

```
const char* const EpicsTpy::EPICS_DB_FORBIDDEN[ ]
```

Initial value:

```
= {"", EPICS_DB_SCAN, "DOL", EPICS_DB_INP, EPICS_DB_OUT, "VAL", "RVAL", "INIT",
     "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
     "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of forbidden EPICS record fields

Definition at line 69 of file TpyToEpicsConst.h.

6.11.2.9 EPICS_DB_HHSV

```
const char* const EpicsTpy::EPICS_DB_HHSV = "HHSV"
```

high severity high limit

Definition at line 54 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.10 EPICS_DB_HIGH

```
const char* const EpicsTpy::EPICS_DB_HIGH = "HIGH"
```

low severity high limit

Definition at line 50 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.11 EPICS_DB_HIHI

```
const char* const EpicsTpy::EPICS_DB_HIHI = "HIHI"
```

high severity high limit

Definition at line 49 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.12 EPICS_DB_HOPR

```
const char* const EpicsTpy::EPICS_DB_HOPR = "HOPR"
```

high ops value string

Definition at line 25 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.13 EPICS_DB_HSV

```
const char* const EpicsTpy::EPICS_DB_HSV = "HSV"
```

low severity high limit

Definition at line 55 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.14 EPICS_DB_HYST

```
const char* const EpicsTpy::EPICS_DB_HYST = "HYST"
```

deadband

Definition at line 53 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.15 EPICS_DB_INP

```
const char* const EpicsTpy::EPICS_DB_INP = "INP"
```

input link

Definition at line 39 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.16 EPICS_DB_LLSV

```
const char* const EpicsTpy::EPICS_DB_LLSV = "LLSV"
```

high severity low limit

Definition at line 57 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.17 EPICS_DB_LOLO

```
const char* const EpicsTpy::EPICS_DB_LOLO = "LOLO"
```

high severity low limit

Definition at line 52 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.18 EPICS_DB_LOPR

```
const char* const EpicsTpy::EPICS_DB_LOPR = "LOPR"
```

low ops value string

Definition at line 26 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.19 EPICS_DB_LOW

```
const char* const EpicsTpy::EPICS_DB_LOW = "LOW"
```

low severity low limit

Definition at line 51 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.20 EPICS_DB_LSV

```
const char* const EpicsTpy::EPICS_DB_LSV = "LSV"
```

low severity low limit

Definition at line 56 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.21 EPICS_DB_MAJOR

```
const char* const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"
```

major alarm

Definition at line 61 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()(), and EpicsTpy::epics_db_processing::process_field←_alarm().

6.11.2.22 EPICS_DB_MINOR

```
const char* const EpicsTpy::EPICS_DB_MINOR = "MINOR"
```

minor alarm

Definition at line 60 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()(), and EpicsTpy::epics_db_processing::process_field←_alarm().

6.11.2.23 EPICS_DB_NOALARM

```
const char* const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"
```

no alarm

Definition at line 59 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::process_field_alarm().

6.11.2.24 EPICS_DB_NUMVAL

```
const char* const EpicsTpy::EPICS_DB_NUMVAL[ ]
```

Initial value:

```
= {EPICS_DB_TSE, EPICS_DB_HOPR, EPICS_DB_Lopr, EPICS_DB_DrvH, EPICS_DB_DrvL, "EGUF", "EGUL",
    "AOFF", "ASLO", "ESLO", "SMOO", "PREC", "HIHI", "LOLO", "HIGH", "LOW", "HYST",
    "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL", "EIVL", "NIVL",
    "TEVL", "ELVL",
    "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of EPICS record fields which are numeric

Definition at line 81 of file TpyToEpicsConst.h.

6.11.2.25 EPICS_DB_ONAM

```
const char* const EpicsTpy::EPICS_DB_ONAM = "ONAM"
```

one name string

Definition at line 29 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.26 EPICS_DB_OSV

```
const char* const EpicsTpy::EPICS_DB_OSV = "OSV"
```

one severity

Definition at line 45 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.27 EPICS_DB_OUT

```
const char* const EpicsTpy::EPICS_DB_OUT = "OUT"
```

output link

Definition at line 40 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.28 EPICS_DB_PINI

```
const char* const EpicsTpy::EPICS_DB_PINI = "PINI"
```

initialization

Definition at line 42 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.29 EPICS_DB_PREC

```
const char* const EpicsTpy::EPICS_DB_PREC = "PREC"
```

precision

Definition at line 32 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.30 EPICS_DB_SCAN

```
const char* const EpicsTpy::EPICS_DB_SCAN = "SCAN"
```

Scan

Definition at line 38 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.31 EPICS_DB_TSE

```
const char* const EpicsTpy::EPICS_DB_TSE = "TSE"
```

time stamp

Definition at line 41 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.32 EPICS_DB_UNSV

```
const char* const EpicsTpy::EPICS_DB_UNSV = "UNSV"
```

unknown severity

Definition at line 63 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.33 EPICS_DB_ZNAM

```
const char* const EpicsTpy::EPICS_DB_ZNAM = "ZNAM"
```

zero name string

Definition at line 30 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.34 EPICS_DB_ZRST

```
const char* const EpicsTpy::EPICS_DB_ZRST[16]
```

Initial value:

```
= {"ZRST", "ONST", "TWST", "THST", "FRST", "FVST", "SXST", "SVST",
     "EIST", "NIST", "TEST", "ELST", "TVST", "TTST", "FTST", "FFST"}
```

enum string

Definition at line 33 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.35 EPICS_DB_ZRSV

```
const char* const EpicsTpy::EPICS_DB_ZRSV[16]
```

Initial value:

```
= {"ZRSV", "ONSV", "TWSV", "THSV", "FRSV", "FVSV", "SXSV", "SVSV",
     "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV", "FTSV", "FFSV"}
```

enum alarm severity

Definition at line 64 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.36 EPICS_DB_ZRVL

```
const char* const EpicsTpy::EPICS_DB_ZRVL[16]
```

Initial value:

```
= {"ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
    "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL"}
```

enum val

Definition at line 35 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.37 EPICS_DB_ZSV

```
const char* const EpicsTpy::EPICS_DB_ZSV = "ZSV"
```

zero severity

Definition at line 46 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.38 LIGODAQ_DATATYPE_DEFAULT

```
const int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT
```

default DAQ datatype is float

Definition at line 91 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.39 LIGODAQ_DATATYPE_FLOAT

```
const int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4
```

DAQ datatype is float

Definition at line 89 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.40 LIGODAQ_DATATYPE_INT32

```
const int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2
```

DAQ datatype is int32

Definition at line 90 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.41 LIGODAQ_DATATYPE_NAME

```
const char* const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"
```

DAQ datatype name

Definition at line 88 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.42 LIGODAQ_INI_HEADER

```
const char* const EpicsTpy::LIGODAQ_INI_HEADER
```

Initial value:

```
=  
"[default]\n"  
"gain=1.00\n"  
"datatype=%i\n"  
"ifoid=0\n"  
"slope=6.1028e-05\n"  
"acquire=3\n"  
"offset=0\n"  
"units=%s\n"  
"dcuid=4\n"  
"datarate=16"
```

DAQ ini file header: substitute defaults

Definition at line 98 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.43 LIGODAQ_UNIT_DEFAULT

```
const char* const EpicsTpy::LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE
```

default DAQ unit is none

Definition at line 95 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.44 LIGODAQ_UNIT_NAME

```
const char* const EpicsTpy::LIGODAQ_UNIT_NAME = "units"
```

DAQ unit name

Definition at line 93 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.45 LIGODAQ_UNIT_NONE

```
const char* const EpicsTpy::LIGODAQ_UNIT_NONE = "none"
```

DAQ unit for no unit

Definition at line 94 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.46 MAX_EPICS_CHANNEL

```
const int EpicsTpy::MAX_EPICS_CHANNEL = 54
```

maximum length of EPICS channel name

Definition at line 18 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.47 MAX_EPICS_DESC

```
const int EpicsTpy::MAX_EPICS_DESC = 40
```

maximum length of EPICS channel description

Definition at line 19 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.48 MAX_EPICS_STRING

```
const int EpicsTpy::MAX_EPICS_STRING = 40
```

maximum length of EPICS strings

Definition at line 20 of file TpyToEpicsConst.h.

6.11.2.49 MAX_EPICS_UNIT

```
const int EpicsTpy::MAX_EPICS_UNIT = 15
```

maximum length of EPICS channel unit string

Definition at line 21 of file TpyToEpicsConst.h.

Chapter 7

Namespace Documentation

7.1 DevTc Namespace Reference

Namespace for TCat device support.

Classes

- struct `devTcDefIn`
Device support input record.
- struct `devTcDeflo`
Device support record.
- struct `devTcDefOut`
device support output record.
- struct `devTcDefWaveformIn`
device support waveform record.
- struct `epics_record_traits`
Epics record traits.
- class `epics_tc_db_processing`
EPICS/TCat db processing.
- class `EpicsInterface`
Epics interface class.
- class `register_devsup`
Device support registration.
- class `tcRegisterTolocShell`
Register TC commands.

Typedefs

- typedef std::tuple< `std::stringcase`, `std::stringcase`, `epics_list_processing` *, `bool` > `filename_rule_list_tuple`
Tuple for filnemae, rule and list processing.
- typedef std::vector< `filename_rule_list_tuple` > `tc_listing_def`
List of tuples for filnemae, rule and list processing.
- typedef std::tuple< `std::stringcase`, `std::stringcase`, `epics_macrofiles_processing` *, `const char` * > `dirname_arg_macro_tuple`
Tuple for directory name, argument and macro list processing.
- typedef std::vector< `dirname_arg_macro_tuple` > `tc_macro_def`
List of tuples for directory name, argument and macro list processing.

Enumerations

- enum `epics_record_enum` {
 aaival = 0, `aaoval`, `aival`, `aoval`,
 `bival`, `boval`, `eventval`, `histogramval`,
 `longinval`, `longoutval`, `mbbival`, `mbboval`,
 `mbbiDirectval`, `mbboDirectval`, `stringinval`, `stringoutval`,
 `waveformval`, `airval`, `aorval`, `birval`,
 `borval`, `mbbiDirectrval`, `mbboDirectrval`, `mbbirval`,
 `mbborval`, `epics_record_enumEnd`, `invalidval` = -1 }

Epics record type enum.

Functions

- const std::regex `tc_regex` ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?) +(8[0-9][0-9]))/(\\d{1,9})/(\\d{1,9})\\:\\\\(\\d{1,9}))")

Callback for output record.
- const std::regex `info_regex` ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?) +(8[0-9][0-9]))/(info)/([A-Za-z0-9_]+))")

Regex for identifying info records.
- void `tcLoadRecords` (const iocshArgBuf *args)

Load TwinCAT records.
- void `tcSetScanRate` (const iocshArgBuf *args)

Set the scan rate.
- void `tcList` (const iocshArgBuf *args)

Generate channel lists.
- void `tcMacro` (const iocshArgBuf *args)

Generated macro files.
- void `tcAlias` (const iocshArgBuf *args)

Define alias and replacement rules.
- void `tcInfoPrefix` (const iocshArgBuf *args)

Sets the info prefix.
- void `tcPrintVals` (const iocshArgBuf *args)

Print all values.
- void `tcPrintVal` (const iocshArgBuf *args)

Print value.

7.1.1 Detailed Description

Namespace for TCat device support.

`DevTc` Name space

`DevTc` namespace

7.1.2 Function Documentation

7.1.2.1 tc_regex()

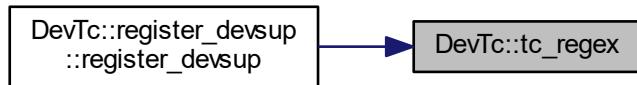
```
const std::regex DevTc::tc_regex (
    "((tc)://((\\b([0-9]|1[0-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?) +(8[0-9][0-9]))/)(\\d{1,9})/(
    :\\d{1,9})"
```

Callback for output record.

Callback function to process EPICS out recordRegex for indentifying TwinCAT records

Referenced by DevTc::register_devsup::register_devsup().

Here is the caller graph for this function:



7.1.2.2 tcAlias()

```
void DevTc::tcAlias (
    const iocshArgBuf * args )
```

Define alias and replacement rules.

Define a nick name or alias

Parameters

<i>args</i>	Arguments for tcAlias
-------------	-----------------------

Definition at line 747 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



7.1.2.3 tcInfoPrefix()

```
void DevTc::tcInfoPrefix (
    const iocshArgBuf * args )
```

Sets the info prefix.

Sets the channel prefix for info PLC records

Parameters

<i>args</i>	Arguments for tcInfoPrefix
-------------	----------------------------

Definition at line 801 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



7.1.2.4 tcList()

```
void DevTc::tcList (
    const iocshArgBuf * args )
```

Generate channel lists.

List function to generate separate listings

Parameters

<i>args</i>	Arguments for tcList
-------------	----------------------

Definition at line 697 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



7.1.2.5 tcLoadRecords()

```
void DevTc::tcLoadRecords (
    const iocshArgBuf * args )
```

Load TwinCAT records.

Function for loading a TCat tpy file, and using it to generate internal record entries as well as the EPICs .db file

Parameters

<i>args</i>	Arguments for tcLoadRecords
-------------	-----------------------------

Definition at line 475 of file drvTc.cpp.

7.1.2.6 tcMacro()

```
void DevTc::tcMacro (
    const iocshArgBuf * args )
```

Generated macro files.

Macro function to generate macro files

Parameters

<i>args</i>	Arguments for tcMacro
-------------	-----------------------

Definition at line 722 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



7.1.2.7 tcPrintVal()

```
void DevTc::tcPrintVal ( const iocshArgBuf * args )
```

Print value.

Debugging function that prints the values for one or multiple records of the PLCs. Supports wildcards.

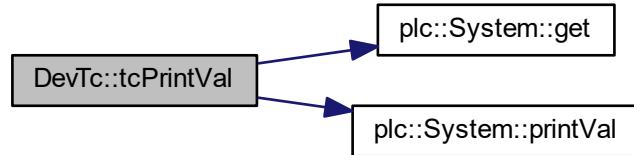
Parameters

<i>args</i>	Arguments for tcPrintVal
-------------	--------------------------

Definition at line 843 of file drvTc.cpp.

References plc::System::get(), and plc::System::printVal().

Here is the call graph for this function:



7.1.2.8 tcPrintVals()

```
void DevTc::tcPrintVals (
    const iocshArgBuf * args )
```

Print all values.

Debugging function that prints the values for all records of the PLCs

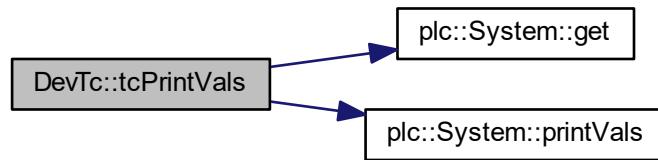
Parameters

<i>args</i>	Arguments for tcPrintVals
-------------	---------------------------

Definition at line 832 of file drvTc.cpp.

References plc::System::get(), and plc::System::printVals().

Here is the call graph for this function:



7.1.2.9 tcSetScanRate()

```
void DevTc::tcSetScanRate (const iocshArgBuf * args )
```

Set the scan rate.

Set scan rate of the read scanner

Parameters

<i>args</i>	Arguments for tcSetScanRate
-------------	-----------------------------

Definition at line 640 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



7.2 EpicsTpy Namespace Reference

Namespace for tpy-db conversion.

Classes

- class `epics_conversion`
Epics conversion.
- class `epics_db_processing`
Epics database record processing
- class `epics_list_processing`
List processing.
- class `epics_macrofiles_processing`
Macro file processing.
- struct `macro_info`
Macro information.
- struct `macro_record`
Macro record.
- class `multi_io_support`
Multiple IO support.
- class `split_io_support`
Split IO support.

TypeDefs

- typedef std::vector< `macro_info` > `macro_list`
- typedef std::stack< `macro_record` > `macro_stack`
- typedef std::unordered_set< std::stringcase > `filename_set`

Enumerations

- enum `tc_epics_conv` { `no_conversion`, `no_dot`, `ligo_std`, `ligo_vac` }
Conversion rules for TC/EPICS.
- enum `case_type` { `preserve_case`, `upper_case`, `lower_case` }
Case conversion rule enum.
- enum `io_filestat` { `io_filestat::closed`, `read`, `write` }
enum for file io
- enum `listing_type` { `listing_standard`, `listing_autoburt`, `listing_daqini` }
Listing type enum.
- enum `macrofile_type` { `macrofile_type::all`, `macrofile_type::fields`, `macrofile_type::errors` }
- enum `device_support_type` { `device_support_opc_name`, `device_support_tc_name` }
Device support enum.

Variables

- const int `MAX_EPICS_CHANNEL` = 54
- const int `MAX_EPICS_DESC` = 40
- const int `MAX_EPICS_STRING` = 40
- const int `MAX_EPICS_UNIT` = 15
- const char *const `EPICS_DB_EGU` = "EGU"
- const char *const `EPICS_DB_DESC` = "DESC"
- const char *const `EPICS_DB_HOPR` = "HOPR"
- const char *const `EPICS_DB_LOPR` = "LOPR"
- const char *const `EPICS_DB_DRVH` = "DRVH"
- const char *const `EPICS_DB_DRVL` = "DRVL"
- const char *const `EPICS_DB_ONAM` = "ONAM"
- const char *const `EPICS_DB_ZNAM` = "ZNAM"
- const char *const `EPICS_DB_PREC` = "PREC"
- const char *const `EPICS_DB_ZRST` [16]
- const char *const `EPICS_DB_ZRVL` [16]
- const char *const `EPICS_DB_SCAN` = "SCAN"
- const char *const `EPICS_DB_INP` = "INP"
- const char *const `EPICS_DB_OUT` = "OUT"
- const char *const `EPICS_DB_TSE` = "TSE"
- const char *const `EPICS_DB_PINI` = "PINI"
- const char *const `EPICS_DB_DTYP` = "DTYP"
- const char *const `EPICS_DB_OSV` = "OSV"
- const char *const `EPICS_DB_ZSV` = "ZSV"
- const char *const `EPICS_DB_COSV` = "COSV"
- const char *const `EPICS_DB_HIHI` = "HIHI"
- const char *const `EPICS_DB_HIGH` = "HIGH"
- const char *const `EPICS_DB_LOW` = "LOW"
- const char *const `EPICS_DB_LOLO` = "LOLO"
- const char *const `EPICS_DB_HYST` = "HYST"
- const char *const `EPICS_DB_HHSV` = "HHSV"
- const char *const `EPICS_DB_HSV` = "HSV"
- const char *const `EPICS_DB_LSV` = "LSV"
- const char *const `EPICS_DB_LLSV` = "LLSV"
- const char *const `EPICS_DB_NOALARM` = "NO_ALARM"
- const char *const `EPICS_DB_MINOR` = "MINOR"
- const char *const `EPICS_DB_MAJOR` = "MAJOR"
- const char *const `EPICS_DB_UNSV` = "UNSV"
- const char *const `EPICS_DB_ZRSV` [16]
- const char *const `EPICS_DB_FORBIDDEN` []
- const char *const `EPICS_DB_ALLOWED` []
- const char *const `EPICS_DB_NUMVAL` []
- const char *const `LIGODAQ_DATATYPE_NAME` = "datatype"
- const int `LIGODAQ_DATATYPE_FLOAT` = 4
- const int `LIGODAQ_DATATYPE_INT32` = 2
- const int `LIGODAQ_DATATYPE_DEFAULT` = `LIGODAQ_DATATYPE_FLOAT`
- const char *const `LIGODAQ_UNIT_NAME` = "units"
- const char *const `LIGODAQ_UNIT_NONE` = "none"
- const char *const `LIGODAQ_UNIT_DEFAULT` = `LIGODAQ_UNIT_NONE`
- const char *const `LIGODAQ_INI_HEADER`

7.2.1 Detailed Description

Namespace for tpy-db conversion.

[EpicsTpy](#) name space

7.3 InfoPlc Namespace Reference

Namespace for Info communication.

Classes

- class [InfoInterface](#)
Info interface.
- class [process_arg_info](#)
Arguments for processing.

Typedefs

- typedef bool([InfoInterface](#)::* [info_update_method](#)) ()
Pointer to info update method.
- typedef std::tuple< [ParseUtil::variable_name](#), [ParseUtil::process_type_enum](#), [ParseUtil::opc_list](#), std::stringcase, bool, [update_enum](#), [info_update_method](#) > [info_dbrecord_type](#)
- typedef std::vector< [info_dbrecord_type](#) > [info_dbrecord_list](#)
List type of db info tuples.

Enumerations

- enum [update_enum](#) { [update_enum::forever](#), [update_enum::once](#), [update_enum::done](#) }
Update frequency type.

7.3.1 Detailed Description

Namespace for Info communication.

[InfoPlc](#) name space, which has all the classes and functions used for communicating with Info.

7.4 ParseTpy Namespace Reference

Namespace for parsing.

Classes

- class [ads_routing_info](#)
ADS routing information.
- class [base_record](#)
Base record definition.
- class [compiler_info](#)
Compiler information.
- class [item_record](#)
item record
- class [parserinfo_type](#)
Parser information.
- class [project_record](#)
Project information.
- class [symbol_record](#)
Symbol record.
- class [tpy_file](#)
Tpy file parsing.
- class [type_map](#)
Type dictionary.
- class [type_record](#)
Type record information.

Typedefs

- typedef std::pair< int, int > [dimension](#)
- typedef std::list< [dimension](#) > [dimensions](#)
- typedef std::map< int, [std::stringcase](#) > [enum_map](#)
- typedef std::pair< int, [std::stringcase](#) > [enum_pair](#)
- typedef std::list< [item_record](#) > [item_list](#)
- typedef std::multimap< unsigned int, [type_record](#) > [type_multimap](#)
- typedef std::list< [symbol_record](#) > [symbol_list](#)

Enumerations

- enum [type_enum](#) {
 unknown, simple, arraytype, enumtype,
 structtype, functionblock
 }
- Type enum.*

Functions

- bool [compareNamesWoNamespace](#) (const [std::stringcase](#) &p1, const [std::stringcase](#) &p2)
- bool [get_decoration](#) (const char **atts, unsigned int &decoration)
- bool [get_pointer](#) (const char **atts)

Variables

- const char *const **xmlPlcProjectInfo** = "PlcProjectInfo"
PLC project info.
- const char *const **xmlProjectInfo** = "ProjectInfo"
Project info.
- const char *const **xmlRoutingInfo** = "RoutingInfo"
Routing info.
- const char *const **xmlCompilerInfo** = "CompilerInfo"
Compiler info.
- const char *const **xmlAdsInfo** = "AdsInfo"
ADS info.
- const char *const **xmlDataTypes** = "DataTypes"
Data types.
- const char *const **xmlDataType** = "DataType"
Data type.
- const char *const **xmlSymbols** = "Symbols"
Symbols.
- const char *const **xmlSymbol** = "Symbol"
Symbol.
- const char *const **xmlProperties** = "Properties"
Properties.
- const char *const **xmlProperty** = "Property"
Property.
- const char *const **xmlCompilerVersion** = "CompilerVersion"
Compiler version.
- const char *const **xmlTwinCATVersion** = "TwinCATVersion"
TwinCAT version.
- const char *const **xmlCpuFamily** = "CpuFamily"
CPU family.
- const char *const **xmlNetId** = "NetId"
Net ID.
- const char *const **xmlPort** = "Port"
Port.
- const char *const **xmlTargetName** = "TargetName"
Target name.
- const char *const **xmlName** = "Name"
Name.
- const char *const **xmlType** = "Type"
Type.
- const char *const **xmlAttrDecoration** = "Decoration"
Decoration.
- const char *const **xmlAttrPointer** = "Pointer"
Pointer.
- const char *const **xmlIGroup** = "IGroup"
I Group.
- const char *const **xmlIOffset** = "IOffset"
I Offset.
- const char *const **xmlBitSize** = "BitSize"
Bit size.
- const char *const **xmlBitOffs** = "BitOffs"

- Bit Offset.*
- const char *const `xmlArrayInfo` = "ArrayInfo"
Array info.
 - const char *const `xmlArrayLBound` = "LBound"
Lower bound.
 - const char *const `xmlArrayElements` = "Elements"
Elements.
 - const char *const `xmlSubItem` = "SubItem"
Sub item.
 - const char *const `xmlFbInfo` = "FbInfo"
Fb info.
 - const char *const `xmlEnumInfo` = "EnumInfo"
Enum info.
 - const char *const `xmlEnumText` = "Text"
Text.
 - const char *const `xmlEnumEnum` = "Enum"
Enum.
 - const char *const `xmlEnumComment` = "Comment"
Comment.
 - const char *const `xmlValue` = "Value"
Value.
 - const char *const `xmlDesc` = "Desc"
Description.
 - const char *const `opcExport` = "opc"
OPC.
 - const char *const `opcProp` = "opc_prop"
OPC property.
 - const char *const `opcBracket` = "["
OPC bracket.

7.4.1 Detailed Description

Namespace for parsing.

Namespace for tpy parsing.

`ParseTpy` name space

7.4.2 Function Documentation

7.4.2.1 compareNamesWoNamespace()

```
bool ParseTpy::compareNamesWoNamespace (
    const std::stringcase & p1,
    const std::stringcase & p2 )
```

`compareNamesWoNamespace`

Definition at line 354 of file `ParseTpy.cpp`.

7.4.2.2 get_decoration()

```
bool ParseTpy::get_decoration (
    const char ** atts,
    unsigned int & decoration )
```

XML get decoration number from attribute

Definition at line 489 of file ParseTpy.cpp.

References `xmlAttrDecoration`.

7.4.2.3 get_pointer()

```
bool ParseTpy::get_pointer (
    const char ** atts )
```

XML get pointer from attribute

Definition at line 503 of file ParseTpy.cpp.

References `xmlAttrPointer`.

7.5 ParseUtil Namespace Reference

Namespace for parsing utilities.

Classes

- class [bit_location](#)
Bit location.
- class [memory_location](#)
Memory location.
- class [opc_list](#)
OPC list.
- class [optarg](#)
Optional arguments.
- class [process_arg](#)
Arguments for processing.
- class [process_arg_tc](#)
Arguments for processing.
- class [replacement_rules](#)
Replacement rules.
- class [tag_processing](#)
Tag processing selection.
- class [variable_name](#)
Variable name.

Typedefs

- `typedef std::map< std::stringcase, std::stringcase > replacement_table`
- `typedef std::map< int, std::stringcase > property_map`
- `typedef std::pair< int, std::stringcase > property_el`

Enumerations

- `enum opc_enum { no_change, publish, silent }`
OPC state enum.
- `enum process_type_enum {`
 `pt_invalid, pt_int, pt_real, pt_bool,`
 `pt_string, pt_enum, pt_binary }`
Process type.
- `enum process_tag_enum { process_all, process_atomic, process_structured }`
Tag preoicessing enum.

Variables

- `const int OPC_PROP_CDT = 1`
- `const int OPC_PROP_VALUE = 2`
- `const int OPC_PROP_QUALITY = 3`
- `const int OPC_PROP_TIME = 4`
- `const int OPC_PROP_RIGHTS = 5`
- `const int OPC_PROP_SCANRATE = 6`
- `const int OPC_PROP_UNIT = 100`
- `const int OPC_PROP_DESC = 101`
- `const int OPC_PROP_HIEU = 102`
- `const int OPC_PROP_LOEU = 103`
- `const int OPC_PROP_HIRANGE = 104`
- `const int OPC_PROP_LORANGE = 105`
- `const int OPC_PROP_CLOSE = 106`
- `const int OPC_PROP_OPEN = 107`
- `const int OPC_PROP_TIMEZONE = 108`
- `const int OPC_PROP_FGC = 201`
- `const int OPC_PROP_BGC = 202`
- `const int OPC_PROP_BLINK = 203`
- `const int OPC_PROP_BMP = 204`
- `const int OPC_PROP SND = 205`
- `const int OPC_PROP_HTML = 206`
- `const int OPC_PROP_AVI = 207`
- `const int OPC_PROP_ALMSTAT = 300`
- `const int OPC_PROP_ALMHELP = 301`
- `const int OPC_PROP_ALMAREAS = 302`
- `const int OPC_PROP_ALMPRIMARYAREA = 303`
- `const int OPC_PROP_ALMCCONDITION = 304`
- `const int OPC_PROP_ALMLIMIT = 305`
- `const int OPC_PROP_ALMDB = 306`
- `const int OPC_PROP_ALMH = 307`
- `const int OPC_PROP_ALMH = 308`
- `const int OPC_PROP_ALML = 309`
- `const int OPC_PROP_ALMLL = 310`

- const int `OPC_PROP_ALMROC` = 311
- const int `OPC_PROP_ALMDEV` = 312
- const int `OPC_PROP_PREC` = 8500
- const int `OPC_PROP_ZRST` = 8510
- const int `OPC_PROP_FFST` = 8525
- const int `OPC_PROP_RECTYPE` = 8600
- const int `OPC_PROP_INOUT` = 8601
- const char *const `OPC_PROP_INPUT` = "input"
- const char *const `OPC_PROP_OUTPUT` = "output"
- const int `OPC_PROP_TSE` = 8602
- const int `OPC_PROP_PINI` = 8603
- const int `OPC_PROP_DTYP` = 8604
- const int `OPC_PROP_SERVER` = 8610
- const int `OPC_PROP_PLCNNAME` = 8611
- const int `OPC_PROP_ALIAS` = 8620
- const int `OPC_PROP_ALMOSV` = 8700
- const int `OPC_PROP_ALMZSV` = 8701
- const int `OPC_PROP_ALMCOSV` = 8702
- const int `OPC_PROP_ALMUNSV` = 8703
- const int `OPC_PROP_ALMZRSV` = 8710
- const int `OPC_PROP_ALMFFSV` = 8725
- const int `OPC_PROP_ALMHHSV` = 8727
- const int `OPC_PROP_ALMHHSV` = 8728
- const int `OPC_PROP_ALMLSV` = 8729
- const int `OPC_PROP_ALMLLSV` = 8730

7.5.1 Detailed Description

Namespace for parsing utilities.

`ParseUtil` name space

`ParseTpy` name space

7.6 plc Namespace Reference

Namespace for abstract plc functionality.

Classes

- class `BasePLC`
Base PLC.
- class `BaseRecord`
Class for managing a tag/channel.
- class `DataValue`
Data value.
- struct `DataValueTraits`
Data value traits.
- struct `DataValueTypeDef`
Collection of type definitions.
- class `Interface`
Abstract interface.
- struct `scanner_thread_args`
Scanner thread arguments.
- class `System`
System to keep track of PLCs.

Typedefs

- `typedef std::shared_ptr< BasePLC > BasePLCPtr`
Smart pointer to PLC.
- `typedef std::unique_ptr< Interface > InterfacePtr`
Smart pointer to interface.
- `typedef std::shared_ptr< BaseRecord > BaseRecordPtr`
smart pointer to record
- `typedef std::unordered_map< std::stringcase, BaseRecordPtr > BaseRecordList`
list of record
- `typedef std::map< std::stringcase, BasePLCPtr > BasePLCList`
BasePLC map.

Enumerations

- `enum data_type_enum {`
`dtInvalid, dtBool, dtInt8, dtUInt8,`
`dtInt16, dtUInt16, dtInt32, dtUInt32,`
`dtInt64, dtUInt64, dtFloat, dtDouble,`
`dtString, dtWString, dtBinary }`
Data type enumeration.
- `enum access_rights_enum { read_only, write_only, read_write }`
Access rights enum.

Functions

- `template<>`
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest,`
`DataValueTypeDef::atomic_string *source)`
Reset and read.
- `template<>`
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool`
`&read_pending, DataValueTypeDef::atomic_bool &valid, DataValueTypeDef::atomic_wstring *dest, const`
`DataValueTypeDef::type_string &source)`
Write and test.
- `VOID CALLBACK ScannerProc (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)`
Scanner thread callback.
- `DWORD WINAPI scannerThread (scanner_thread_args args)`
Scanner thread.
- `template<typename T, typename U >`
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, U source)`
Reset and read.
- `template<typename T >`
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, typename DataValueTraits< T >::traits_atomic *source)`
Reset and read.
- `template<typename T, typename U >`
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending,`
`DataValueTypeDef::atomic_bool &valid, U dest, const T &source)`
Write and test.
- `template<typename T >`
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending,`
`DataValueTypeDef::atomic_bool &valid, typename DataValueTraits< T >::traits_atomic *dest,`
`const T &source)`
Write and test.

7.6.1 Detailed Description

Namespace for abstract plc functionality.

PLC namespace

7.6.2 Typedef Documentation

7.6.2.1 BasePLCList

```
typedef std::map<std::stringcase, BasePLCPtr> plc::BasePLCList
```

[BasePLC](#) map.

This is list of [BasePLC](#), ordered by their name.

Definition at line 893 of file plcBase.h.

7.6.2.2 BasePLCPtr

```
typedef std::shared_ptr<BasePLC> plc::BasePLCPtr
```

Smart pointer to PLC.

This is a smart pointer to a PLC

Definition at line 20 of file plcBase.h.

7.6.2.3 BaseRecordList

```
typedef std::unordered_map<std::stringcase, BaseRecordPtr> plc::BaseRecordList
```

list of record

This is a list of tag/channel records organized as a hash map

Definition at line 710 of file plcBase.h.

7.6.2.4 BaseRecordPtr

```
typedef std::shared_ptr<BaseRecord> plc::BaseRecordPtr
```

smart pointer to record

This is a smart pointer to a tag/channel record

Definition at line 705 of file plcBase.h.

7.6.2.5 InterfacePtr

```
typedef std::unique_ptr<Interface> plc::InterfacePtr
```

Smart pointer to interface.

This is a smart pointer for [Interface](#)

Definition at line 69 of file plcBase.h.

7.6.3 Enumeration Type Documentation

7.6.3.1 access_rights_enum

```
enum plc::access_rights_enum
```

Access rights enum.

Enum for access rights of a record

Enumerator

read_only	Read only.
write_only	Write only.
read_write	Read/write.

Definition at line 479 of file plcBase.h.

7.6.3.2 data_type_enum

```
enum plc::data_type_enum
```

Data type enumeration.

This is an enumerated type listing all the available data types

Enumerator

dtInvalid	Invalid data type.
dtBool	Boolean.
dtInt8	1-byte integer
dtUInt8	1-byte unsigned integer
dtInt16	2-byte integer
dtUInt16	2-byte unsigned integer
dtInt32	4-byte integer
dtUInt32	4-byte unsigned integer
dtInt64	8-byte integer
dtUInt64	8-byte unsigned integer
dtFloat	4-byte single precision floating point
dtDouble	8-byte double precision floating point
dtString	string class
dtWString	wstring class
dtBinary	binary object

Definition at line 74 of file plcBase.h.

7.6.4 Function Documentation

7.6.4.1 reset_and_read() [1/3]

```
template<typename T , typename U >
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    T & dest,
    U source )
```

Reset and read.

Will read the value and reset the dirty flag.

Definition at line 21 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.6.4.2 reset_and_read() [2/3]

```
template<typename T >
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    T & dest,
    typename DataValueTraits< T >::traits_atomic * source )
```

Reset and read.

Will read the value and reset the dirt flag (same type).

Definition at line 34 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.6.4.3 reset_and_read() [3/3]

```
template<>
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    DataValueTypeDef::type_wstring & dest,
    DataValueTypeDef::atomic_string * source )
```

Reset and read.

Will read the value and reset the dirty flag (wstring from string).

Definition at line 73 of file plcBase.cpp.

References plc::DataValueTypeDef::memory_order.

Referenced by plc::DataValue::Read().

Here is the caller graph for this function:



7.6.4.4 ScannerProc()

```
VOID CALLBACK plc::ScannerProc (
    LPVOID lpArg,
    DWORD dwTimerLowValue,
    DWORD dwTimerHighValue )
```

Scanner thread callback.

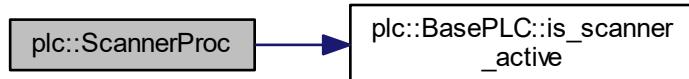
Scanner thread callback with periodic timer

Definition at line 673 of file plcBase.cpp.

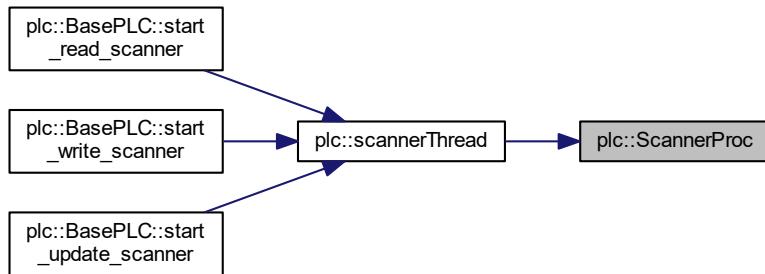
References plc::BasePLC::is_scanner_active(), plc::scanner_thread_args::plc, and plc::scanner_thread_args::scanner.

Referenced by scannerThread().

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.4.5 scannerThread()

```
DWORD WINAPI plc::scannerThread (
    scanner_thread_args args )
```

Scanner thread.

Scanner thread with periodic timer This function uses the windows waitable timer which will call a completion routine at a regular interval. The completion routine in this case is one of either read_scanner, write_scanner, or update_scanner.

Definition at line 692 of file plcBase.cpp.

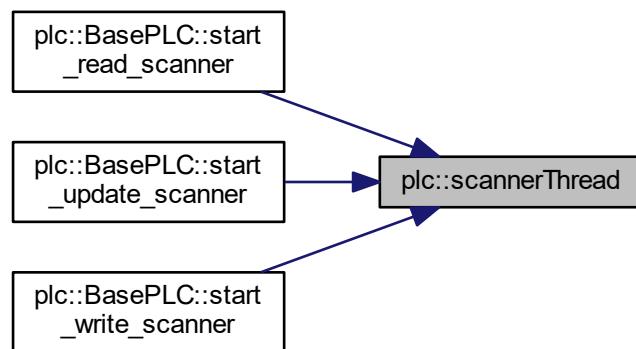
References ScannerProc(), and plc::scanner_thread_args::scanperiod.

Referenced by plc::BasePLC::start_read_scanner(), plc::BasePLC::start_update_scanner(), and plc::BasePLC::start_write_scanner().

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.4.6 write_and_test() [1/3]

```

template<typename T, typename U >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    U dest,
    const T & source )
  
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one.

Definition at line 48 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.6.4.7 `write_and_test()` [2/3]

```
template<typename T >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    typename DataValueTraits< T >::traits_atomic * dest,
    const T & source )
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one (same type).

Definition at line 68 of file plcBaseTemplate.h.

References `plc::DataValueTypeDef::memory_order`.

7.6.4.8 `write_and_test()` [3/3]

```
template<>
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    DataValueTypeDef::atomic_wstring * dest,
    const DataValueTypeDef::type_string & source )
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one (string to wstring).

Definition at line 90 of file plcBase.cpp.

References `plc::DataValueTypeDef::memory_order`.

Referenced by `plc::DataValue::Write()`.

Here is the caller graph for this function:



7.7 TcComms Namespace Reference

Namespace for TCat communication.

Classes

- class [AmsRouterNotification](#)
AMS Router Notification.
- struct [DataPar](#)
Memory location struct.
- class [TCatInterface](#)
TCat interface class.
- class [TcPLC](#)
TwinCAT PLC.
- class [tcProcWrite](#)
TwinCAT process write requests.

Functions

- void [errorPrintf](#) (int nErr)
errorPrintf
- void __stdcall [ADScallback](#) (AmsAddr *pAddr, AdsNotificationHeader *pNotification, unsigned long plcId)
- void __stdcall [RouterCall](#) (long nReason)

Variables

- const int [MAX_REQ_SIZE](#) = 250000
maximum allowed request size (bytes)
- const int [MAX_SINGLE_GAP_SIZE](#) = 50
maximum allowed size (bytes) of a memory gap within continuous request
- const double [MAX_REL_GAP](#) = 0.25
(maximum allowed total gap size) / (current request size)
- const int [MIN_REL_GAP_SIZE](#) = 100
minimum allowed relative gap size (bytes)
- const int [default_scanrate](#) = 100
default PLC TwinCAT scan rate (100ms)
- const int [minimum_scanrate](#) = 5
minimum PLC TwinCAT scan rate (5ms)
- const int [maximum_scanrate](#) = 10000
maximum PLC TwinCAT scan rate (10s)
- const int [default_multiple](#) = 10
default multiple for PLC EPICS scan rate (10)
- const int [minimum_multiple](#) = 1
minimum multiple for PLC EPICS scan rate (1)
- const int [maximum_multiple](#) = 200
maximum multiple for PLC EPICS scan rate (200)

7.7.1 Detailed Description

Namespace for TCat communication.

[TcComms](#) name space, which has all the classes and functions used for communicating with TCat.

7.7.2 Function Documentation

7.7.2.1 ADScallback()

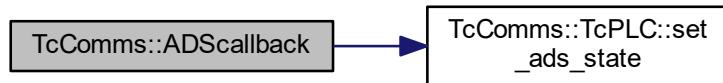
```
void __stdcall TcComms::ADScallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId )
```

Callback for ADS state change

Definition at line 648 of file tcComms.cpp.

References TcComms::TcPLC::set_ads_state().

Here is the call graph for this function:



7.7.2.2 errorPrintf()

```
void TcComms::errorPrintf (
    int nErr )
```

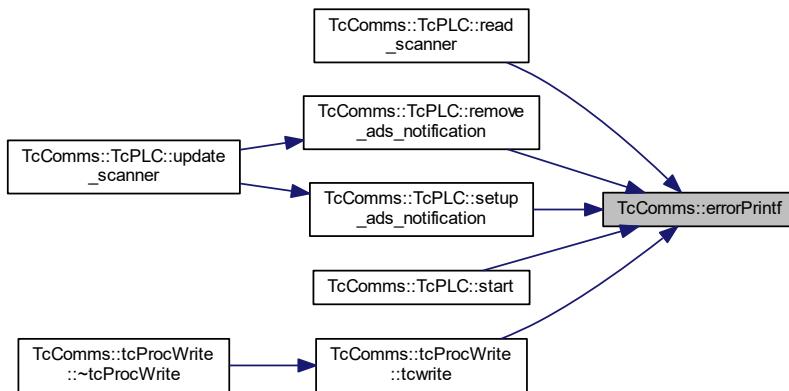
errorPrintf

Print an error message for an ADS error return code

Definition at line 31 of file tcComms.cpp.

Referenced by TcComms::TcPLC::read_scanner(), TcComms::TcPLC::remove_ads_notification(), TcComms::TcPLC::setup_ads_notification(), TcComms::TcPLC::start(), and TcComms::tcProcWrite::tcwrite().

Here is the caller graph for this function:



7.7.2.3 RouterCall()

```
void __stdcall TcComms::RouterCall (
    long nReason )
```

Callback for AMS router state change

Definition at line 877 of file tcComms.cpp.

Chapter 8

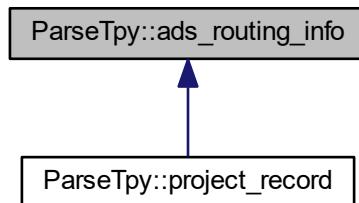
Class Documentation

8.1 ParseTpy::ads_routing_info Class Reference

ADS routing information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::ads_routing_info:



Public Member Functions

- `ads_routing_info ()`
Default constructor.
- `ads_routing_info (const std::string& netid, int port=801)`
Constructor.
- `ads_routing_info (const std::string& netid, int port, const std::string& targetname)`
Constructor.
- `const std::string& get_netid () const`
Get ADS net id.
- `void set_netid (const std::string& netid)`
Set ADS net id.
- `int get_port () const`

- `void set_port (int port)`
Set ADS port.
- `const std::stringcase & get_targetname () const`
Get ADS target name.
- `void set_targetname (const std::stringcase &targetname)`
Set ADS target name.
- `bool isValid () const`
Checks, if net id is of the form n.n.n.n.n.n.
- `std::stringcase get () const`
- `bool set (const std::stringcase &s)`
- `bool get (unsigned char &a1, unsigned char &a2, unsigned char &a3, unsigned char &a4, unsigned char &a5, unsigned char &a6) const`

Protected Attributes

- `std::stringcase ads_netid`
ADS net ID.
- `int ads_port`
ADS port.
- `std::stringcase ads_targetname`
ADS target name.

8.1.1 Detailed Description

ADS routing information.

This is a base class for storing the ADS routing information

Definition at line 23 of file ParseTpy.h.

8.1.2 Member Function Documentation

8.1.2.1 `get()` [1/2]

```
std::stringcase ParseTpy::ads_routing_info::get ( ) const
```

Gets a string representation of a ads routing information

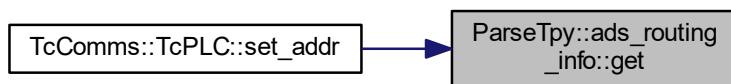
Returns

string with format "tc://netid:port/", empty on error

Definition at line 215 of file ParseTpy.cpp.

Referenced by TcComms::TcPLC::set_addr().

Here is the caller graph for this function:

**8.1.2.2 get() [2/2]**

```
bool ParseTpy::ads_routing_info::get (
    unsigned char & a1,
    unsigned char & a2,
    unsigned char & a3,
    unsigned char & a4,
    unsigned char & a5,
    unsigned char & a6 ) const
```

Get address in net format

Parameters

<i>a1</i>	First address qualifier (return)
<i>a2</i>	Second address qualifier (return)
<i>a3</i>	Third address qualifier (return)
<i>a4</i>	Fourth address qualifier (return)
<i>a5</i>	Fifth address qualifier (return)
<i>a6</i>	Sixth address qualifier (return)

Definition at line 228 of file ParseTpy.cpp.

8.1.2.3 set()

```
bool ParseTpy::ads_routing_info::set (
    const std::string& s )
```

Set the ads routing information using a string of the form: "tc://netid:port/" where netid is a string of the format n.n.n.n.n

Parameters

s	String describing the ads routing information
---	---

Returns

True if successful

Definition at line 249 of file ParseTpy.cpp.

The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.2 TcComms::AmsRouterNotification Class Reference

AMS Router Notification.

```
#include <tcComms.h>
```

Public Member Functions

- int [get_ads_version \(\) const](#)
get ADS protocol/library version
- int [get_ads_revision \(\) const](#)
get ADS protocol/library revision
- int [get_ads_build \(\) const](#)
get ADS protocol/library build

Static Public Member Functions

- static AmsRouterEvent [get_router_notification \(\)](#)
get router notification
- static const [AmsRouterNotification & get_instance \(\)](#)
get global instance

Protected Attributes

- int [ads_version](#)
ADS protocol/library version.
- int [ads_revision](#)
ADS protocol/library revision.
- int [ads_build](#)
ADS protocol/library build.

Friends

- void __stdcall RouterCall (long)
Notification callback is a friend.

8.2.1 Detailed Description

AMS Router Notification.

Class for a AMS router notifications

Definition at line 409 of file tcComms.h.

8.2.2 Friends And Related Function Documentation

8.2.2.1 RouterCall

```
void __stdcall RouterCall (
    long nReason ) [friend]
```

Notification callback is a friend.

Callback for AMS router state change

Definition at line 877 of file tcComms.cpp.

The documentation for this class was generated from the following files:

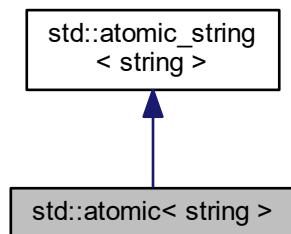
- [tcComms.h](#)
- [tcComms.cpp](#)

8.3 std::atomic< string > Class Template Reference

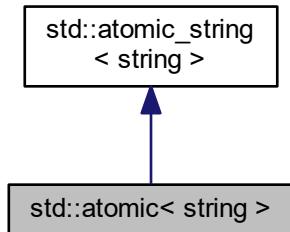
[atomic<string>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< string >:



Collaboration diagram for std::atomic< string >:



Public Member Functions

- [atomic \(\)](#)
Default constructor.
- [atomic \(const string &s\)](#)
Constructor from string.

Additional Inherited Members

8.3.1 Detailed Description

```
template<>
class std::atomic< string >
```

[atomic<string>](#)

This is a class implements an atomic specialization for std::string

Definition at line 71 of file [atomic_string.h](#).

The documentation for this class was generated from the following file:

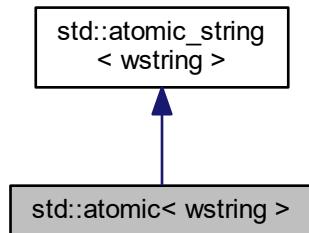
- [atomic_string.h](#)

8.4 std::atomic< wstring > Class Template Reference

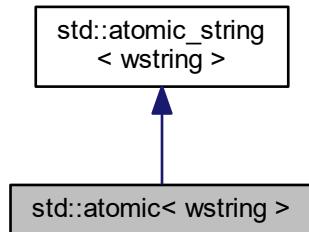
[atomic<wstring>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< wstring >:



Collaboration diagram for std::atomic< wstring >:



Public Member Functions

- [atomic \(\)](#)
Default constructor.
- [atomic \(const wstring &s\)](#)
Constructor from string.

Additional Inherited Members

8.4.1 Detailed Description

```
template<>
class std::atomic< wstring >
```

[atomic<wstring>](#)

This is a class implements an atomic specialization for std::wstring

Definition at line 84 of file [atomic_string.h](#).

The documentation for this class was generated from the following file:

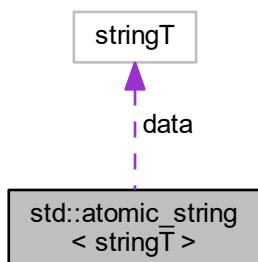
- [atomic_string.h](#)

8.5 std::atomic_string< stringT > Class Template Reference

atomic strings

```
#include <atomic_string.h>
```

Collaboration diagram for std::atomic_string< stringT >:



Public Member Functions

- [atomic_string \(\)](#)
Default constructor.
- [atomic_string \(const stringT &s\)](#)
Constructor from data.
- [stringT operator= \(const stringT &right\)](#)
Assignment operator on basic type.
- [bool is_lock_free \(\) const](#)
Not lock free.
- [void store \(stringT value, memory_order order=memory_order_seq_cst\)](#)
Store.
- [stringT load \(memory_order order=memory_order_seq_cst\) const](#)
Load.
- [operator stringT \(\) const](#)
Convert to string.
- [stringT exchange \(const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Exchange.
- [bool compare_exchange_weak \(stringT &exp, const stringT &value, memory_order order1, memory_order order2\)](#)
Compare exchange.
- [bool compare_exchange_weak \(stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Compare exchange.
- [bool compare_exchange_strong \(stringT &exp, const stringT &value, memory_order order1, memory_order order2\)](#)
Compare exchange.
- [bool compare_exchange_strong \(stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Compare exchange.

Protected Attributes

- [atomic_flag flag](#)
flag for spin lock
- [stringT data](#)
data string

8.5.1 Detailed Description

```
template<typename stringT>
class std::atomic_string< stringT >
```

atomic strings

This is a class implements an atomic specialization for strings

Definition at line 16 of file atomic_string.h.

8.5.2 Member Function Documentation

8.5.2.1 operator=()

```
template<typename stringT>
stringT std::atomic_string< stringT >::operator= (
    const stringT & right )
```

Assignment operator on basic type.

Assignment operator.

Definition at line 96 of file atomic_string.h.

The documentation for this class was generated from the following file:

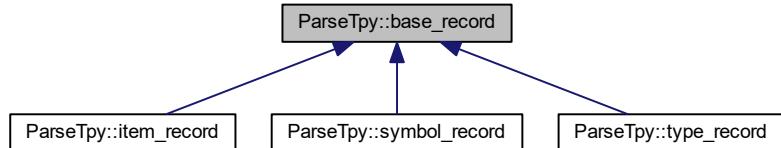
- [atomic_string.h](#)

8.6 ParseTpy::base_record Class Reference

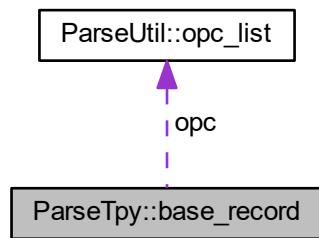
Base record definition.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::base_record:



Collaboration diagram for ParseTpy::base_record:



Public Member Functions

- `base_record ()`
Default constructor.
- `base_record (const std::stringcase &n)`
- `base_record (const std::stringcase &n, const ParseUtil::opc_list &o)`
- `base_record (const std::stringcase &n, const ParseUtil::opc_list &o, const std::stringcase &tn, int td=0)`
- `base_record (const std::stringcase &n, const std::stringcase &tn, int td=0)`
- `const std::stringcase & get_name () const`
Get name.
- `std::stringcase & get_name ()`
Get name.
- `void set_name (std::stringcase n)`
Set name.
- `const std::stringcase & get_type_name () const`
Get type name.
- `std::stringcase & get_type_name ()`
Get type name.
- `void set_type_name (std::stringcase t)`
Set type name.
- `unsigned int get_type_decoration () const`
Get type decoration.
- `void set_type_decoration (unsigned int id)`
Set type decoration.
- `bool get_type_pointer () const`
Get type pointer.
- `void set_type_pointer (bool isPointer)`
Set type pointer.
- `const ParseUtil::opc_list & get_opc () const`
Get OPC list.
- `ParseUtil::opc_list & get_opc ()`
Get OPC list.

Protected Attributes

- `std::stringcase name`
name of type
- `std::stringcase type_n`
type definition
- `unsigned int type_decoration`
decoration or type ID of type definition
- `bool type_pointer`
this is a pointer
- `ParseUtil::opc_list opc`
list of opc properties

8.6.1 Detailed Description

Base record definition.

This is a base class for storing name, type, type id and opc list

Definition at line 149 of file ParseTpy.h.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 base_record() [1/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n ) [inline], [explicit]
```

Constructor

Parameters

<i>n</i>	Name
----------	------

Definition at line 156 of file ParseTpy.h.

8.6.2.2 base_record() [2/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o ) [inline]
```

Constructor

Parameters

<i>n</i>	Name
<i>o</i>	OPC list

Definition at line 161 of file ParseTpy.h.

8.6.2.3 base_record() [3/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o,
    const std::stringcase & tn,
    int td = 0 ) [inline]
```

Constructor

Parameters

<i>n</i>	Name
<i>o</i>	OPC list
<i>tn</i>	Type name
<i>td</i>	Type decortation or id

Definition at line 168 of file ParseTpy.h.

8.6.2.4 base_record() [4 / 4]

```
ParseTpy::base_record::base_record (
    const std::string& n,
    const std::string& tn,
    int td = 0 ) [inline]
```

Constructor

Parameters

<i>n</i>	Name
<i>tn</i>	Type name
<i>td</i>	Type decortation or id

Definition at line 175 of file ParseTpy.h.

The documentation for this class was generated from the following file:

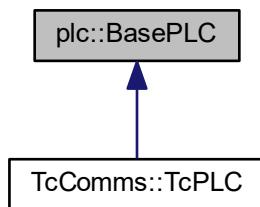
- [ParseTpy.h](#)

8.7 plc::BasePLC Class Reference

Base PLC.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BasePLC:



Public Types

- `typedef std::recursive_mutex mutex_type`
Defines the mutex type.
- `typedef std::lock_guard< mutex_type > guard`
Defined the mutex guard type.
- `typedef DataValueTypeDef::type_uint64 time_type`
Define timestamp type.
- `typedef void(BasePLC::* scanner_func) ()`
Function pointer to scanner.

Public Member Functions

- `BasePLC ()`
Default constructor.
- `virtual ~BasePLC ()`
Destructor.
- `int get_read_scanner_period () const`
Get read scanner period in ms.
- `void set_read_scanner_period (int period)`
Set read scanner period in ms.
- `bool start_read_scanner ()`
Start read scanner.
- `bool terminate_read_scanner ()`
Terminate read scanner.
- `int get_write_scanner_period () const`
Get write scanner period in ms.
- `void set_write_scanner_period (int period)`
Set write scanner period in ms.
- `bool start_write_scanner ()`
Start write scanner.
- `bool terminate_write_scanner ()`
Terminate write scanner.
- `int get_update_scanner_period () const`
Get update scanner period in ms.
- `void set_update_scanner_period (int period)`
Set update scanner period in ms.
- `bool start_update_scanner ()`
Start update scanner.
- `bool terminate_update_scanner ()`
Terminate update scanner.
- `bool is_scanner_active () const`
is scanner active?
- `void set_scanners_active (bool active)`
set scanner active state
- `void reserve (BaseRecordList::size_type n)`
- `bool add (BaseRecord *precord)`
- `bool add (BaseRecordPtr precord)`
- `BaseRecordPtr find (const std::stringcase &name)`
- `bool erase (const std::stringcase &name)`
- `bool get_next (BaseRecordPtr &next, const BaseRecordPtr &prev) const`

- template<typename func >
void **for_each** (func &f)
- int **count** () const
Count the number of records.
- virtual void **printAllRecords** ()
Print all records and vals to stdout. (override for action)
- virtual void **printRecord** (const std::string &var)
- virtual **time_type get_timestamp** () const
Get time stamp.
- virtual **time_t get_timestamp_unix** () const
- virtual void **set_timestamp** (**time_type** tstamp)
Set time stamp.
- virtual void **update_timestamp** ()
Set time stamp to current time.
- const **std::stringcase & get_name** () const
Get name.
- const **std::stringcase & get_alias** () const
Get nick name/alias.
- void **set_alias** (const **std::stringcase &nickname**)
Set nick name/alias.
- virtual bool **start** ()
- virtual void **user_data_set_valid** (bool valid)
- virtual void **plc_data_set_valid** (bool valid)

Protected Member Functions

- void **set_name** (const **std::stringcase &n**)
Set name (careful! This is used for indexing in the PLCList of System)
- virtual void **read_scanner** ()
read scanner (override for action)
- virtual void **write_scanner** ()
write scanner (override for action)
- virtual void **update_scanner** ()
update scanner (override for action)

Protected Attributes

- **mutex_type mux**
Mutex to synchronize access to this class.
- **std::stringcase name**
Name.
- **std::stringcase alias**
Nick name or alias (used to generate info record names)
- **BaseRecordList records**
- **time_type timestamp**
Time stamp.
- int **read_scanner_period**
read scanner period in ms
- int **write_scanner_period**
write scanner period in ms

- int `update_scanner_period`
update scanner period in ms
- std::atomic< bool > `scanners_active`
scanners are active
- std::thread `read_thread`
read thread
- std::thread `write_thread`
write thread
- std::thread `update_thread`
update thread

8.7.1 Detailed Description

Base PLC.

This is a base class for interfacing a programmable logic controller. It contains and manages a list of tag/channel records. This is a base class which needs to be used a derived class by a real implementation.

This class is MT safe and uses a mutex to synchronize access.

Definition at line 720 of file plcBase.h.

8.7.2 Member Function Documentation

8.7.2.1 `add()` [1/2]

```
bool plc::BasePLC::add (
    BaseRecord * precord ) [inline]
```

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

<code>precord</code>	Pointer to record. Will be adopted
----------------------	------------------------------------

Returns

true, if it could be added

Definition at line 784 of file plcBase.h.

8.7.2.2 add() [2/2]

```
bool plc::BasePLC::add (
```

```
    BaseRecordPtr precord )
```

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

<i>precord</i>	Smart pointer to record.
----------------	--------------------------

Returns

true, if it could be added

Definition at line 547 of file plcBase.cpp.

References mux, and records.

8.7.2.3 erase()

```
bool plc::BasePLC::erase (
    const std::stringcase & name )
```

Erase a tag/channel record.

Parameters

<i>name</i>	Name of record
-------------	----------------

Returns

true if erased

Definition at line 575 of file plcBase.cpp.

References name, and records.

8.7.2.4 find()

```
BaseRecordPtr plc::BasePLC::find (
    const std::stringcase & name )
```

Find a new tag/channel record.

Parameters

<i>name</i>	Name of record
-------------	----------------

Returns

Smart pointer to record (contains nullptr when not found)

Definition at line 561 of file plcBase.cpp.

References mux, name, and records.

8.7.2.5 for_each()

```
template<typename func >
void plc::BasePLC::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

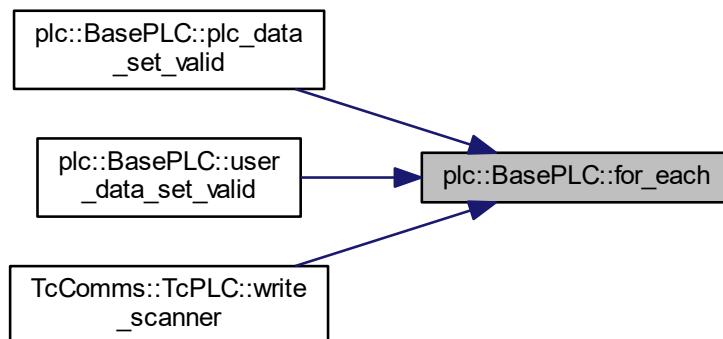
<i>f</i>	Function which takes BaseRecord* as the argument
----------	--

Definition at line 205 of file plcBaseTemplate.h.

References mux, and records.

Referenced by plc_data_set_valid(), user_data_set_valid(), and TcComms::TcPLC::write_scanner().

Here is the caller graph for this function:



8.7.2.6 get_next()

```
bool plc::BasePLC::get_next (
    BaseRecordPtr & next,
    const BaseRecordPtr & prev ) const
```

Get next record in list. Resets to the beginning, if the list changes between subsequent access. This is an MT safe access method to cycle through the list. However, it is not high performance and is meant for slow external access.

Parameters

<i>next</i>	Next tag/channel record (return)
<i>prev</i>	tag/channel record

Returns

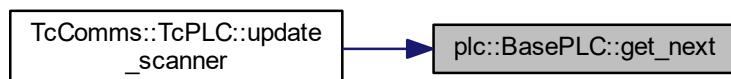
true if successful

Definition at line 583 of file plcBase.cpp.

References mux, and records.

Referenced by TcComms::TcPLC::update_scanner().

Here is the caller graph for this function:

**8.7.2.7 get_timestamp_unix()**

```
time_t plc::BasePLC::get_timestamp_unix ( ) const [virtual]
```

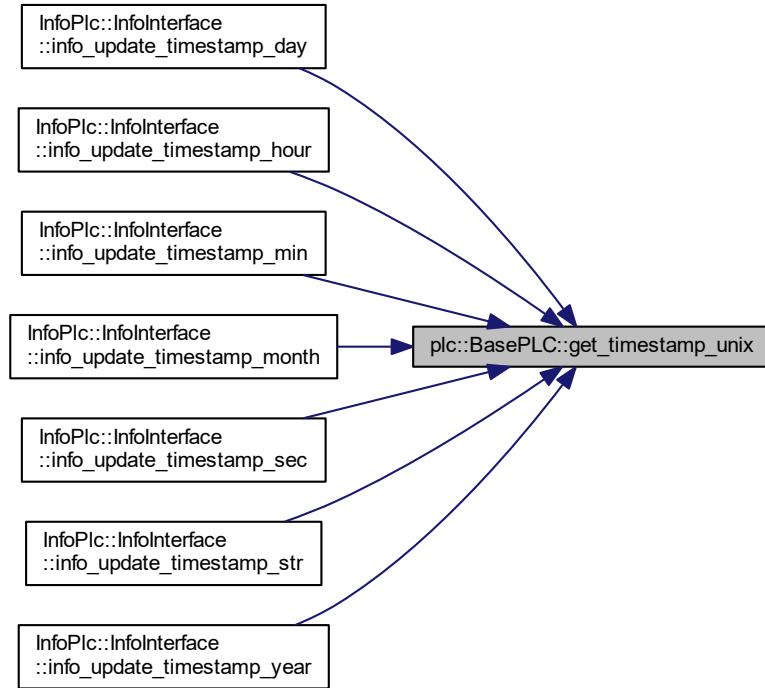
Get time stamp as unix time (seconds since 1970-01-01 00:00:00) Does not include leap seconds

Definition at line 607 of file plcBase.cpp.

References timestamp.

Referenced by InfoPlc::InfoInterface::info_update_timestamp_day(), InfoPlc::InfoInterface::info_update_timestamp_hour(), InfoPlc::InfoInterface::info_update_timestamp_min(), InfoPlc::InfoInterface::info_update_timestamp_month(), InfoPlc::InfoInterface::info_update_timestamp_sec(), InfoPlc::InfoInterface::info_update_timestamp_str(), and InfoPlc::InfoInterface::info_update_timestamp_year().

Here is the caller graph for this function:



8.7.2.8 plc_data_set_valid()

```
void plc::BasePLC::plc_data_set_valid (
    bool valid ) [virtual]
```

Set the valid flag for all data values by the plc

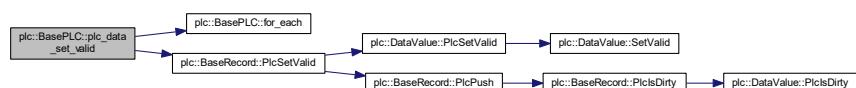
Parameters

<code>valid</code>	Valid flag, true for valid, false for invalid
--------------------	---

Definition at line 649 of file plcBase.cpp.

References for_each(), and plc::BaseRecord::PlcSetValid().

Here is the call graph for this function:



8.7.2.9 printRecord()

```
virtual void plc::BasePLC::printRecord (
    const std::string & var ) [inline], [virtual]
```

Print a record values to stdout. (override for action)

Parameters

<i>var</i>	variable name (accepts wildcards)
------------	-----------------------------------

Reimplemented in [TcComms::TcPLC](#).

Definition at line 819 of file plcBase.h.

8.7.2.10 reserve()

```
void plc::BasePLC::reserve (
    BaseRecordList::size_type n ) [inline]
```

Reserves the given number of elements in the tag/channel list. Use this function when you know many elements are added beforehand to avoid unnecessary rehashing.

Parameters

<i>n</i>	Number of expected tag/channel records
----------	--

Definition at line 779 of file plcBase.h.

References records.

8.7.2.11 start()

```
virtual bool plc::BasePLC::start ( ) [inline], [virtual]
```

This function is called by the driver support in [drvtc.cpp](#), after the PLC has been initialized. This function is overridden by the derived PLCs, and generally will start all the scanner threads.

Returns

true if successful

Reimplemented in [TcComms::TcPLC](#).

Definition at line 843 of file plcBase.h.

8.7.2.12 user_data_set_valid()

```
void plc::BasePLC::user_data_set_valid (
    bool valid ) [virtual]
```

Set the valid flag for all data values by the user

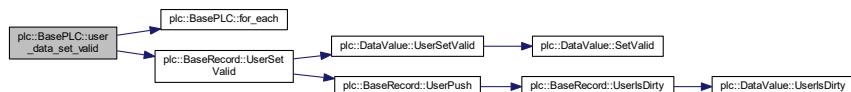
Parameters

<code>valid</code>	Valid flag, true for valid, false for invalid
--------------------	---

Definition at line 639 of file plcBase.cpp.

References for_each(), and plc::BaseRecord::UserSetValid().

Here is the call graph for this function:



8.7.3 Member Data Documentation

8.7.3.1 records

`BaseRecordList` plc::BasePLC::records [protected]

List of tags/channels. The load factor is initialized to 0.5.

Definition at line 864 of file plcBase.h.

Referenced by add(), BasePLC(), count(), erase(), find(), for_each(), get_next(), TcComms::TcPLC::printAll<- Records(), TcComms::TcPLC::printRecord(), TcComms::TcPLC::read_scanner(), reserve(), TcComms::TcPLC<- :start(), and TcComms::TcPLC::update_scanner().

The documentation for this class was generated from the following files:

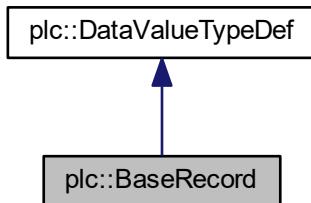
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.8 plc::BaseRecord Class Reference

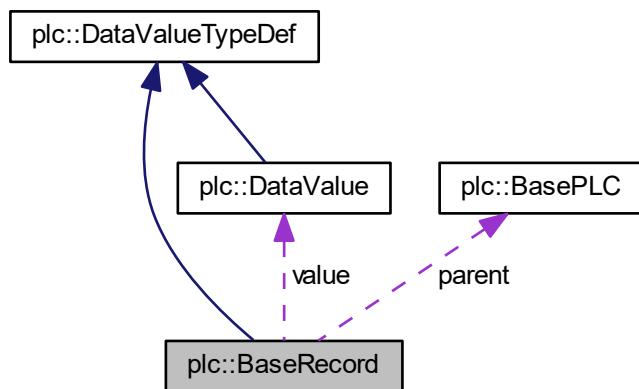
Class for managing a tag/channel.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BaseRecord:



Collaboration diagram for plc::BaseRecord:



Public Member Functions

- [BaseRecord \(\)](#)
Default constructor.
- [BaseRecord \(const std::string& tag\)](#)
- [BaseRecord \(const std::string& recordName, data_type_enum rt, Interface *puser=nullptr, Interface *pplc=nullptr\)](#)
- [virtual ~BaseRecord \(\)](#)
Destructor.

- const `std::stringcase & get_name () const`
Get name.
- void `set_name (const std::stringcase &recordName)`
Set name.
- bool `get_process () const`
Get process flag: false = disabled, true = enabled.
- void `set_process (bool isEnabled)`
Set process flag.
- `access_rights_enum get_access_rights ()`
Get access rights.
- void `set_access_rights (access_rights_enum rights)`
Set access rights.
- `time_type get_timestamp () const`
Get time stamp.
- virtual `Interface * get_userInterface () const`
Get pointer to user interface (no ownership transfer)
- virtual `Interface * get_plcInterface () const`
Get pointer to plc interface (no ownership transfer)
- void `set_userInterface (Interface *puser)`
Set user interface (object will be adopted!)
- void `set_plcInterface (Interface *pplc)`
Set plc interface (object will be adopted!)
- virtual `BasePLC * get_parent () const`
Get parent plc.
- virtual void `set_parent (BasePLC *pPLC)`
Set parent plc.
- const `DataValue & get_data () const`
Get a const reference to the data object.
- `DataValue & get_data ()`
Get a reference to the data object.
- bool `DatasValid ()`
Returns true, if the data is valid.
- template<typename T >
 bool `UserRead (T &data)`
- bool `UserRead (type_string_value *data, size_type max)`
- bool `UserRead (DataValue::type_wstring_value *data, size_type max)`
- template<typename T >
 bool `UserWrite (const T &data)`
- bool `UserWrite (const type_string_value *data, size_type max)`
- bool `UserWrite (const type_wstring_value *data, size_type max)`
- `size_type UserReadBinary (type_binary p, size_type len)`
- `size_type UserWriteBinary (const type_binary p, size_type len)`
- bool `UserIsDirty () const`
Checks if the user needs to read an updated value.
- void `UserSetDirty ()`
Set dirty flag for user.
- bool `UserPush (bool force=false)`
- bool `UserPull ()`
Pulls the user for new data.
- void `UserSetValid (bool valid)`
- bool `UserGetValid ()`

- template<typename T >
bool **PlcRead** (T &data)
- bool **PlcRead** (type_string_value *data, size_type max)
- bool **PlcRead** (type_wstring_value *data, size_type max)
- template<typename T >
bool **PlcWrite** (const T &data)
- bool **PlcWrite** (const type_string_value *data, size_type max)
- bool **PlcWrite** (const type_wstring_value *data, size_type max)
- size_type **PlcReadBinary** (type_binary p, size_type len)
- size_type **PlcWriteBinary** (const type_binary p, size_type len)
- bool **PlcIsDirty** () const
Checks if the plc needs to read an updated value.
- void **PlcSetDirty** ()
Set dirty flag for plc.
- bool **PlcPush** (bool force=false)
- bool **PlcPull** ()
Pulls the plc for new data.
- void **PlcSetValid** (bool valid)
- bool **PlcGetValid** ()

Protected Attributes

- std::stringcase name
Name.
- access_rights_enum access
Enum for access rights.
- atomic_bool process
Process flag: false = disabled, true = enabled.
- **DataValue** value
Data value.
- InterfacePtr plc
PLC interface (master)
- InterfacePtr user
User interface (slave)
- BasePLC * parent
PLC that this record belongs to.

Additional Inherited Members

8.8.1 Detailed Description

Class for managing a tag/channel.

This is the base class for a tag/channel. It contains a data value and owns two pointers to a user and plc interface, respectively. This is the basic work horse for exchanging data between a user (slave) and a plc (master).

Definition at line 495 of file plcBase.h.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 BaseRecord() [1/2]

```
plc::BaseRecord::BaseRecord (
    const std::string& tag ) [inline], [explicit]
```

Constructor

Parameters

<i>tag</i>	Name of tag/channel
------------	---------------------

Definition at line 502 of file plcBase.h.

8.8.2.2 BaseRecord() [2/2]

```
plc::BaseRecord::BaseRecord (
    const std::string& recordName,
    data_type_enum rt,
    Interface * puser = nullptr,
    Interface * pplc = nullptr ) [inline]
```

Constructor

Parameters

<i>recordName</i>	Name of tag/channel
<i>rt</i>	Data type
<i>puser</i>	Pointer to user interface object (will be adopted!)
<i>pplc</i>	Pointer to plc interface object (will be adopted!)

Definition at line 509 of file plcBase.h.

8.8.3 Member Function Documentation

8.8.3.1 PlcGetValid()

```
bool plc::BaseRecord::PlcGetValid () [inline]
```

Get the plc valid flag and reset the dirty flag

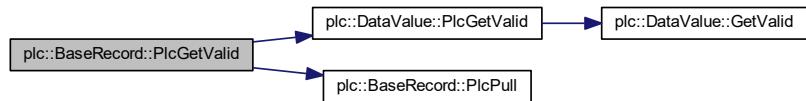
Returns

valid True for valid data, False for invalid

Definition at line 682 of file plcBase.h.

References `plc::DataValue::PlcGetValid()`, `PlcPull()`, `process`, and `value`.

Here is the call graph for this function:

**8.8.3.2 PlcPush()**

```
bool plc::BaseRecord::PlcPush (
    bool force = false ) [inline]
```

Initiated a plc pull, if the data needs an update

Parameters

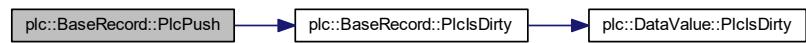
<code>force</code>	Forces a plc update even when not needed
--------------------	--

Definition at line 179 of file plcBaseTemplate.h.

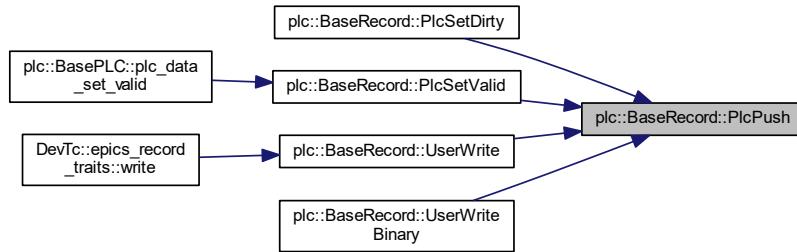
References `PlcIsDirty()`.

Referenced by `PlcSetDirty()`, `PlcSetValid()`, `UserWrite()`, and `UserWriteBinary()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.3 PlcRead() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcRead (
    T & data ) [inline]
```

Execute a plc read, but pull user first

Parameters

<code>data</code>	Reference to data (return)
-------------------	----------------------------

Returns

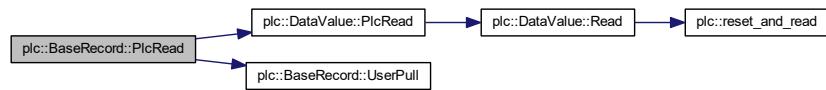
true if successfull

Definition at line 621 of file `plcBase.h`.

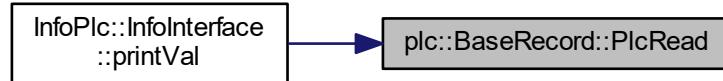
References `plc::DataValue::PlcRead()`, `UserPull()`, and `value`.

Referenced by `InfoPlc::InfoInterface::printVal()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.4 PlcRead() [2/3]

```
bool plc::BaseRecord::PlcRead (
    type_string_value * data,
    size_type max ) [inline]
```

Execute a plc read, but pull user first

Parameters

<i>data</i>	character pointer, pchar (return)
<i>max</i>	Maximum number of characters

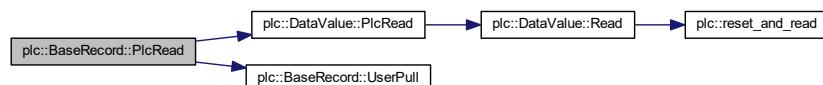
Returns

true if successfull

Definition at line 627 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

Here is the call graph for this function:



8.8.3.5 PlcRead() [3/3]

```
bool plc::BaseRecord::PlcRead (
    type_wstring_value * data,
    size_type max ) [inline]
```

Execute a plc read, but pull user first

Parameters

<i>data</i>	character pointer, pwchar (return)
<i>max</i>	Maximum number of characters

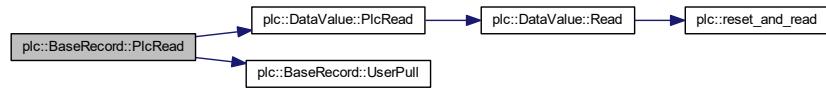
Returns

true if successfull

Definition at line 633 of file plcBase.h.

References `plc::DataValue::PlcRead()`, `UserPull()`, and `value`.

Here is the call graph for this function:

**8.8.3.6 PlcReadBinary()**

```

size_type plc::BaseRecord::PlcReadBinary (
    type_binary p,
    size_type len ) [inline]
  
```

Execute a plc read, but pull user first

Parameters

<i>p</i>	Pointer to data (destination buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes read (0 on error)

Definition at line 658 of file plcBase.h.

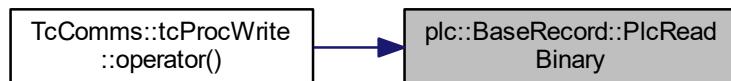
References `plc::DataValue::PlcReadBinary()`, `UserPull()`, and `value`.

Referenced by `TcComms::tcProcWrite::operator()()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.7 PlcSetValid()

```
void plc::BaseRecord::PlcSetValid (
    bool valid ) [inline]
```

Set the plc valid flag and set the dirty flag when flag changes

Parameters

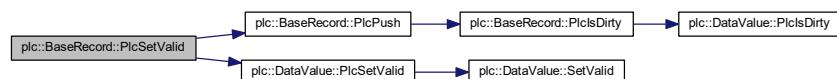
<i>valid</i>	True for valid data, False for invalid
--------------	--

Definition at line 679 of file plcBase.h.

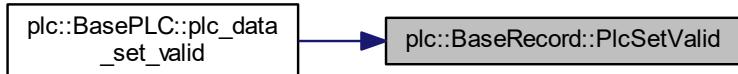
References PlcPush(), plc::DataValue::PlcSetValid(), and value.

Referenced by plc::BasePLC::plc_data_set_valid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.8 PlcWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcWrite (
    const T & data ) [inline]
```

Execute a plc write and push user

Parameters

<i>data</i>	Reference to data
-------------	-------------------

Returns

true if successfull

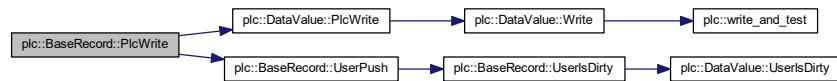
Definition at line 639 of file plcBase.h.

References plc::DataValue::PlcWrite(), UserPush(), and value.

Referenced by InfoPlc::InfoInterface::info_update_active(), InfoPlc::InfoInterface::info_update_ads_build(), InfoPlc::InfoInterface::info_update_ads_netid_b0(), InfoPlc::InfoInterface::info_update_ads_netid_b1(), InfoPlc::InfoInterface::info_update_ads_netid_b2(), InfoPlc::InfoInterface::info_update_ads_netid_b3(), InfoPlc::InfoInterface::info_update_ads_netid_b4(), InfoPlc::InfoInterface::info_update_ads_netid_b5(), InfoPlc::InfoInterface::info_update_ads_netid_str(), InfoPlc::InfoInterface::info_update_ads_port(), InfoPlc::InfoInterface::info_update_ads_revision(), InfoPlc::InfoInterface::info_update_ads_version(), InfoPlc::InfoInterface::info_update_alias(), InfoPlc::InfoInterface::info_update_callback_queue0_free(), InfoPlc::InfoInterface::info_update_callback_queue0_percent(), InfoPlc::InfoInterface::info_update_callback_queue0_size(), InfoPlc::InfoInterface::info_update_callback_queue0_used(), InfoPlc::InfoInterface::info_update_callback_queue1_free(), InfoPlc::InfoInterface::info_update_callback_queue1_percent(), InfoPlc::InfoInterface::info_update_callback_queue1_size(), InfoPlc::InfoInterface::info_update_callback_queue1_used(), InfoPlc::InfoInterface::info_update_callback_queue2_free(), InfoPlc::InfoInterface::info_update_callback_queue2_percent(), InfoPlc::InfoInterface::info_update_callback_queue2_size(), InfoPlc::InfoInterface::info_update_callback_queue2_used(), InfoPlc::InfoInterface::info_update_name(), InfoPlc::InfoInterface::info_update_rate_read(), InfoPlc::InfoInterface::info_update_rate_update(), InfoPlc::InfoInterface::info_update_rate_write(), InfoPlc::InfoInterface::info_update_records_num(), InfoPlc::InfoInterface::info_update_state(), InfoPlc::InfoInterface::info_update_statestr(), InfoPlc::InfoInterface::info_update_svn_local(), InfoPlc::InfoInterface::info_update_svn_revision(), InfoPlc::InfoInterface::info_update_svn_time(), InfoPlc::InfoInterface::info_update_timestamp_day(), InfoPlc::InfoInterface::info_update_timestamp_hour(), InfoPlc::InfoInterface::info_update_timestamp_min(), InfoPlc::InfoInterface::info_update_timestamp_month(), InfoPlc::InfoInterface::info_update_timestamp_sec(), InfoPlc::InfoInterface::info_update_timestamp_str(), InfoPlc::InfoInterface::info_update_timestamp_year(), InfoPlc::InfoInterface::info_update_tpy_filename(), InfoPlc::InfoInterface::info_update_tpy_time_day(), InfoPlc::InfoInterface::info_update_tpy_time_hour(), InfoPlc::InfoInterface::info_update_tpy_time_min(),

`InfoPlc::InfoInterface::info_update_tpy_time_month()`, `InfoPlc::InfoInterface::info_update_tpy_time_sec()`, `InfoPlc::InfoInterface::info_update_tpy_time_str()`, `InfoPlc::InfoInterface::info_update_tpy_time_year()`, and `InfoPlc::InfoInterface::info_update_tpy_valid()`.

Here is the call graph for this function:



8.8.3.9 PlcWrite() [2/3]

```
bool plc::BaseRecord::PlcWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Execute a plc write and push user

Parameters

<code>data</code>	character pointer, pchar
<code>max</code>	Maximum number of characters

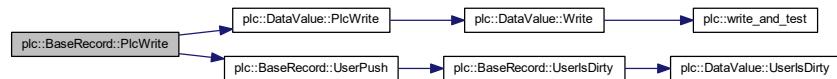
Returns

true if successfull

Definition at line 645 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:



8.8.3.10 PlcWrite() [3/3]

```
bool plc::BaseRecord::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Execute a plc write and push user

Parameters

<i>data</i>	character pointer, pwchar
<i>max</i>	Maximum number of characters

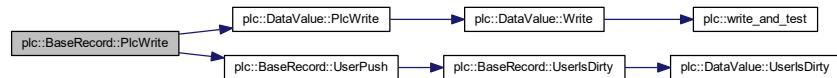
Returns

true if successfull

Definition at line 651 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:

**8.8.3.11 PlcWriteBinary()**

```
size_type plc::BaseRecord::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Execute a plc write and push user

Parameters

<i>p</i>	Pointer to data (source buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes written (0 on error)

Definition at line 664 of file plcBase.h.

References `plc::DataValue::PlcWriteBinary()`, `UserPush()`, and `value`.

Referenced by `TcComms::TcPLC::read_scanner()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.12 UserGetValid()

```
bool plc::BaseRecord::UserGetValid ( ) [inline]
```

Get the user valid flag and reset the dirty flag

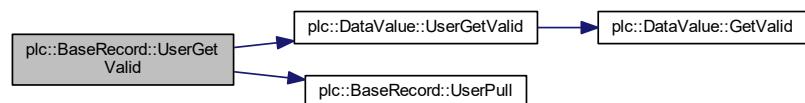
Returns

valid True for valid data, False for invalid

Definition at line 615 of file `plcBase.h`.

References `process`, `plc::DataValue::UserGetValid()`, `UserPull()`, and `value`.

Here is the call graph for this function:



8.8.3.13 UserPush()

```
bool plc::BaseRecord::UserPush (
    bool force = false ) [inline]
```

Initiated a user pull, if the data needs an update

Parameters

<code>force</code>	Forces a user update even when not needed
--------------------	---

Definition at line 153 of file plcBaseTemplate.h.

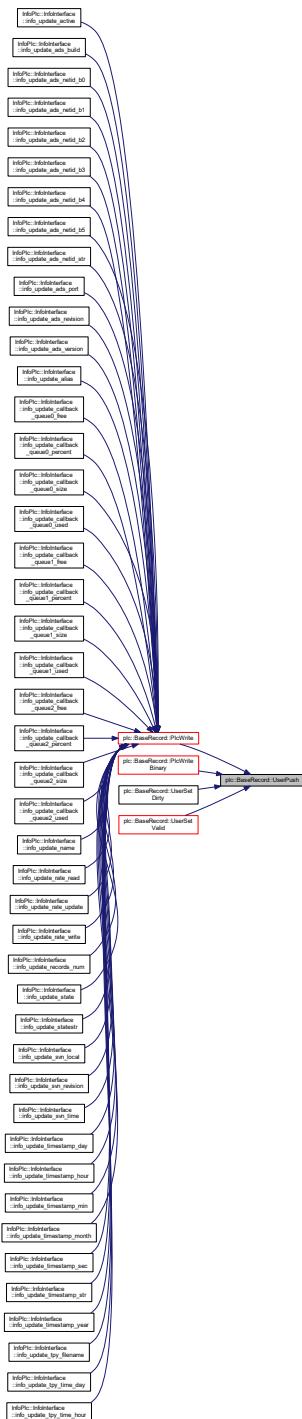
References user, and UserIsDirty().

Referenced by PlcWrite(), PlcWriteBinary(), UserSetDirty(), and UserSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.14 UserRead() [1/3]

```
template<typename T >  
bool plc::BaseRecord::UserRead (
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	Reference to data (return)
-------------	----------------------------

Returns

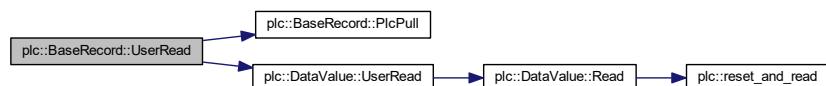
true if successfull

Definition at line 555 of file plcBase.h.

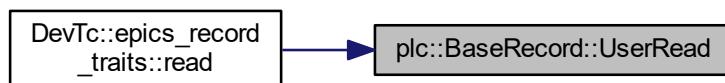
References PlcPull(), plc::DataValue::UserRead(), and value.

Referenced by DevTc::epics_record_traits< RecType >::read().

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.15 UserRead() [2/3]

```

bool plc::BaseRecord::UserRead (
    type_string_value * data,
    size_type max ) [inline]
  
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	character pointer, pchar (return)
<i>max</i>	Maximum number of characters

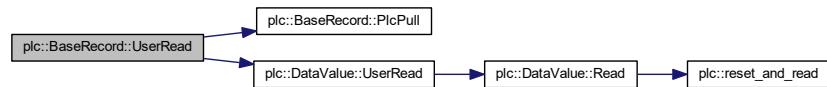
Returns

true if successfull

Definition at line 561 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:

**8.8.3.16 UserRead() [3/3]**

```
bool plc::BaseRecord::UserRead (
    DataValue::type_wstring_value * data,
    size_type max ) [inline]
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	character pointer, pwchar (return)
<i>max</i>	Maximum number of characters

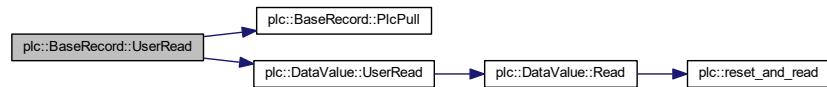
Returns

true if successfull

Definition at line 567 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:



8.8.3.17 UserReadBinary()

```
size_type plc::BaseRecord::UserReadBinary (
    type_binary p,
    size_type len ) [inline]
```

Execute a user read, but pull plc first

Parameters

<i>p</i>	Pointer to data (destination buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes read (0 on error)

Definition at line 591 of file plcBase.h.

References PlcPull(), plc::DataValue::UserReadBinary(), and value.

Here is the call graph for this function:



8.8.3.18 UserSetValid()

```
void plc::BaseRecord::UserSetValid (
    bool valid ) [inline]
```

Set the user valid flag and set the dirty flag when flag changes

Parameters

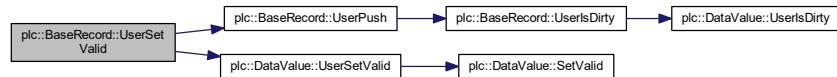
<i>valid</i>	True for valid data, False for invalid
--------------	--

Definition at line 612 of file plcBase.h.

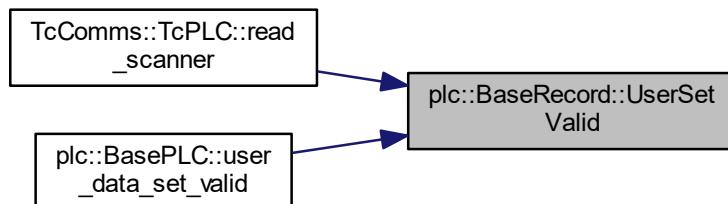
References UserPush(), plc::DataValue::UserSetValid(), and value.

Referenced by TcComms::TcPLC::read_scanner(), and plc::BasePLC::user_data_set_valid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.19 UserWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::UserWrite (
    const T & data ) [inline]
```

Execute a user write and push plc

Parameters

<i>data</i>	Reference to data
-------------	-------------------

Returns

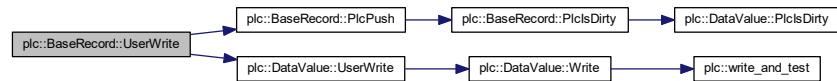
true if successfull

Definition at line 572 of file plcBase.h.

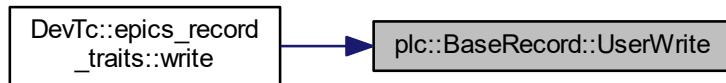
References PlcPush(), plc::DataValue::UserWrite(), and value.

Referenced by DevTc::epics_record_traits< RecType >::write().

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.3.20 UserWrite() [2/3]

```

bool plc::BaseRecord::UserWrite (
    const type_string_value * data,
    size_type max ) [inline]
  
```

Execute a user write and push plc

Parameters

<i>data</i>	character pointer, pchar
<i>max</i>	Maximum number of characters

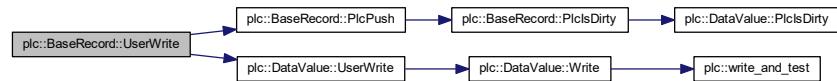
Returns

true if successfull

Definition at line 578 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Here is the call graph for this function:



8.8.3.21 UserWrite() [3/3]

```
bool plc::BaseRecord::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Execute a user write and push plc

Parameters

<i>data</i>	character pointer, pwchar
<i>max</i>	Maximum number of characters

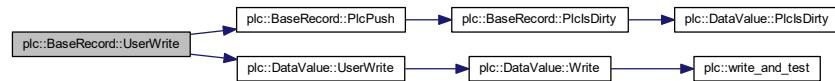
Returns

true if successfull

Definition at line 584 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Here is the call graph for this function:



8.8.3.22 UserWriteBinary()

```
size_type plc::BaseRecord::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Execute a user write and push plc

Parameters

<i>p</i>	Pointer to data (source buffer)
<i>len</i>	Length in bytes (must be the same as data length)

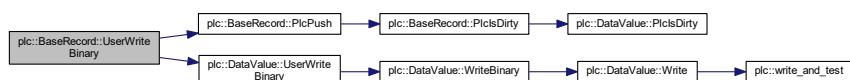
Returns

Number of bytes written (0 on error)

Definition at line 597 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWriteBinary(), and value.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

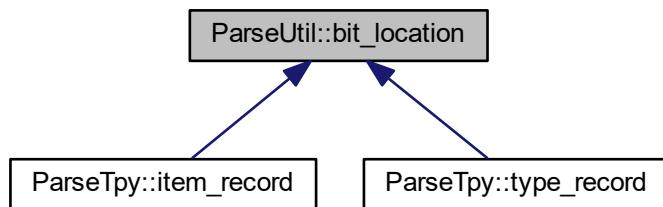
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.9 ParseUtil::bit_location Class Reference

Bit location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::bit_location:



Public Member Functions

- [bit_location \(\)](#)
Default constructor.
- [bit_location \(int bo, int bs\)](#)
Constructor.
- [bool isValid \(\) const](#)
Validity.
- [const int get_bit_offset \(\) const](#)
Get bit offset.
- [void set_bit_offset \(int ofs\)](#)
Set bit offset.
- [const int get_bit_size \(\) const](#)
Get bit size.
- [void set_bit_size \(int size\)](#)
Set bit size.

Protected Attributes

- [int bitoffs](#)
bit offset where elements is stored
- [int bitsize](#)
size in number of bits of symbol

8.9.1 Detailed Description

Bit location.

This is a class for storing bit offset and size in a structure

Definition at line 233 of file ParseUtil.h.

The documentation for this class was generated from the following file:

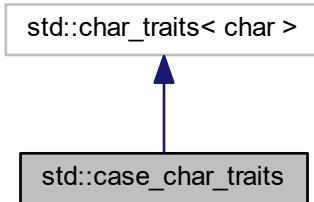
- [ParseUtil.h](#)

8.10 std::case_char_traits Struct Reference

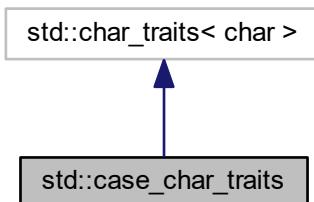
case insensitive traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case_char_traits:



Collaboration diagram for std::case_char_traits:



Static Public Member Functions

- static bool [eq](#) (const char_type &c1, const char_type &c2)
- static bool [ne](#) (const char_type &c1, const char_type &c2)
- static bool [lt](#) (const char_type &c1, const char_type &c2)
- static int [compare](#) (const char_type *s1, const char_type *s2, size_t n)

8.10.1 Detailed Description

case insensitive traits.

This traits class is not case sensitive.

Definition at line 40 of file stringcase.h.

The documentation for this struct was generated from the following file:

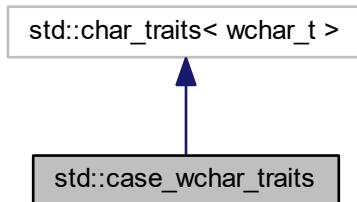
- [stringcase.h](#)

8.11 std::case_wchar_traits Struct Reference

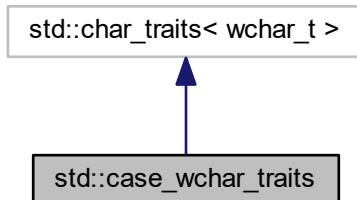
case insensitive unicode traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case_wchar_traits:



Collaboration diagram for std::case_wchar_traits:



Static Public Member Functions

- static bool [eq](#) (const char_type &c1, const char_type &c2)
- static bool [ne](#) (const char_type &c1, const char_type &c2)
- static bool [lt](#) (const char_type &c1, const char_type &c2)
- static int [compare](#) (const char_type *s1, const char_type *s2, size_t n)

8.11.1 Detailed Description

case insensitive unicode traits.

This unicode traits class is not case sensitive.

Definition at line 72 of file stringcase.h.

The documentation for this struct was generated from the following file:

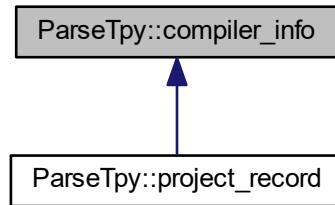
- [stringcase.h](#)

8.12 ParseTpy::compiler_info Class Reference

Compiler information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::compiler_info:



Public Member Functions

- `compiler_info ()`
Default constructor.
- `const std::stringcase & get_cmpl_versionstr () const`
Get compiler version string.
- `void set_cmpl_versionstr (const std::stringcase &versionstr)`
Set compiler version string.
- `double get_cmpl_version () const`
Get compiler version.
- `const std::stringcase & get_tcat_versionstr () const`
Get twincat version string.
- `void set_tcat_versionstr (const std::stringcase &versionstr)`
Set twincat version string.
- `unsigned int get_tcat_version_major () const`
Get twincat major version.
- `unsigned int get_tcat_version_minor () const`
Get twincat minor version.
- `unsigned int get_tcat_version_build () const`
Get twincat build version.
- `const std::stringcase & get_cpu_family () const`
Get cpu family string.
- `void set_cpu_family (const std::stringcase &family)`
Set cpu family string.
- `bool is_cmpl_Valid () const`
Checks, if version is of the form n.n...
- `bool is_tcat_Valid () const`
Checks, if twincat version is of the form n.n...

Protected Attributes

- `std::string``case cmpl_versionstr`
version string
- `double cmpl_version`
version number
- `std::string``case tcat_versionstr`
twincat version string
- `unsigned int tcat_version_major`
twincat major version number
- `unsigned int tcat_version_minor`
twincat minor version number
- `unsigned int tcat_version_build`
twincat build version number
- `std::string``case cpu_family`
cpu family string

8.12.1 Detailed Description

Compiler information.

This is a base class for storing the compiler information

Definition at line 80 of file ParseTpy.h.

The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.13 TcComms::DataPar Struct Reference

Memory location struct.

```
#include <tcComms.h>
```

Public Attributes

- `unsigned long indexGroup`
index group in ADS server
- `unsigned long indexOffset`
index offset in ADS server
- `unsigned long length`
count of bytes to read

8.13.1 Detailed Description

Memory location struct.

Struct for storing index group, index offset, and size of a TC symbol

Definition at line 57 of file tcComms.h.

The documentation for this struct was generated from the following file:

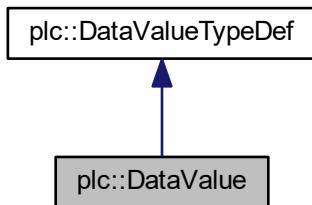
- [tcComms.h](#)

8.14 plc::DataValue Class Reference

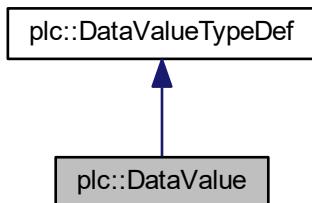
Data value.

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValue:



Collaboration diagram for plc::DataValue:



Public Types

- `typedef void * data_type`
Internally used storage pointer type.

Public Member Functions

- `DataValue ()`
Default constructor.
- `DataValue (data_type_enum rt, size_type len=0)`
- `~DataValue ()`
Destructor.
- `DataValue (const DataValue &)`
Copy constructor.
- `DataValue & operator= (const DataValue &)`
Assignment operator.
- `void Init (data_type_enum rt, size_type len=0)`
- `bool IsValid () const`
is valid
- `data_type_enum get_data_type () const`
get type
- `size_type get_size () const`
get size
- template<typename T >
`bool UserRead (T &data) const`
- template<size_type N>
`bool UserRead (type_string_value(&data)[N]) const`
- `bool UserRead (type_string_value *data, size_type max) const`
- `bool UserRead (type_wstring_value *data, size_type max) const`
- template<typename T >
`bool UserWrite (const T &data)`
- template<size_type N>
`bool UserWrite (const type_string_value(&data)[N])`
- `bool UserWrite (const type_string_value *data, size_type max)`
- `bool UserWrite (const type_wstring_value *data, size_type max)`
- `size_type UserReadBinary (type_binary p, size_type len) const`
- `size_type UserWriteBinary (const type_binary p, size_type len)`
- `bool UserIsDirty () const`
New data for user.
- `void UserSetDirty ()`
Set dirty flag for user.
- `void UserSetValid (bool valid)`
- `bool UserGetValid () const`
- template<typename T >
`bool PlcRead (T &data) const`
- template<size_type N>
`bool PlcRead (type_string_value(&data)[N]) const`
- `bool PlcRead (type_string_value *data, size_type max) const`
- `bool PlcRead (type_wstring_value *data, size_type max) const`
- template<typename T >
`bool PlcWrite (const T &data)`
- template<size_type N>
`bool PlcWrite (const type_string_value(&data)[N])`

- `bool PlcWrite (const type_string_value *data, size_type max)`
- `bool PlcWrite (const type_wstring_value *data, size_type max)`
- `size_type PlcReadBinary (type_binary p, size_type len) const`
- `size_type PlcWriteBinary (const type_binary p, size_type len)`
- `bool PlcIsDirty () const`
New data for plc.
- `void PlcSetDirty ()`
Set dirty flag for plc.
- `void PlcSetValid (bool valid)`
- `bool PlcGetValid () const`

Protected Member Functions

- `DataValue (const DataValue &&)`
Constructor (hidden)
- `DataValue & operator= (const DataValue &&)`
Assignment operator.
- `template<typename T >`
`bool Read (atomic_bool &dirty, T &data) const`
- `bool Read (atomic_bool &dirty, type_string &data) const`
- `bool Read (atomic_bool &dirty, type_wstring &data) const`
- `bool Read (atomic_bool &dirty, type_string_value *data, size_type max) const`
- `bool Read (atomic_bool &dirty, type_wstring_value *data, size_type max) const`
- `template<typename T >`
`bool Write (atomic_bool &dirty, const atomic_bool &pend, const T &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string_value *data, size_type max)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring_value *data, size_type max)`
- `size_type ReadBinary (atomic_bool &dirty, type_binary p, size_type len) const`
- `size_type WriteBinary (atomic_bool &dirty, const atomic_bool &pend, const type_binary p, size_type len)`
- `void SetValid (atomic_bool &dirty, bool valid)`
- `bool GetValid (atomic_bool &dirty) const`

Protected Attributes

- `data_type mydata`
Data pointer.
- `size_type mysize`
- `data_type_enum mytype`
Data type.
- `atomic_bool myvalid`
Valid flag.
- `atomic_bool myuserdirty`
Dirty flag indicating user needs to update.
- `atomic_bool myplcdirty`
Dirty flag indicating plc needs to update.

Additional Inherited Members

8.14.1 Detailed Description

Data value.

Class for data value This class stores a data value and provides synchronization between the user (slave) and the plc (master) interfaces. When the plc writes a value, it is marked dirty on the user side. Then, when the user reads this data, it resets the dirty flag. The same logic applies for writes by the user and reads by the plc.

Data access is guaranteed to be atomic and MT safe. Construction, initialization and destruction is not MT safe and all data access has to be stopped during these operations.

Type conversion is provided between all simple data types. However, the loss of information is not checked upon a down cast. Strings have to be read as strings. Binary data needs to be accessed with the binary read/write operations. However, all data can be accessed through binary access.

Definition at line 223 of file plcBase.h.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 DataValue()

```
plc::DataValue::DataValue (
    data_type_enum rt,
    size_type len = 0 ) [inline], [explicit]
```

Constructor

Parameters

<i>rt</i>	Data type enumeration value
<i>len</i>	Length of data

Definition at line 235 of file plcBase.h.

References Init().

Here is the call graph for this function:



8.14.3 Member Function Documentation

8.14.3.1 GetValid()

```
bool plc::DataValue::GetValid (
    atomic_bool & dirty ) const [protected]
```

Get the valid flag and reset the dirty flag

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
--------------	---------------------------------------

Returns

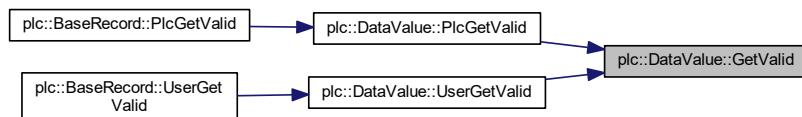
valid True for valid data, False for invalid

Definition at line 506 of file plcBase.cpp.

References plc::DataValueTypeDef::memory_order, and myvalid.

Referenced by PlcGetValid(), and UserGetValid().

Here is the caller graph for this function:



8.14.3.2 Init()

```
void plc::DataValue::Init (
    data_type_enum rt,
    size_type len = 0 )
```

Initializes data value

Parameters

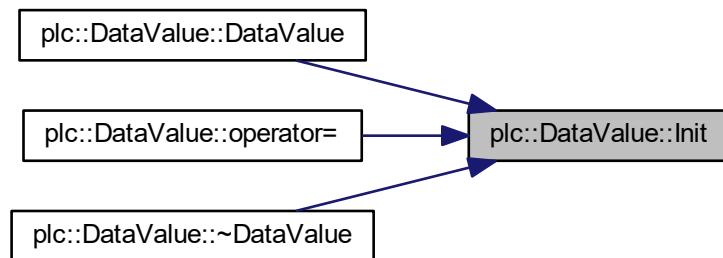
<i>rt</i>	Data type enumeration value
<i>len</i>	Length of data (use only for binary)

Definition at line 189 of file plcBase.cpp.

References `plc::dtBinary`, `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `plc::dtWString`, `mydata`, `myplcdirty`, `mysize`, `mytype`, and `myuserdirty`.

Referenced by `DataValue()`, `operator=()`, and `~DataValue()`.

Here is the caller graph for this function:



8.14.3.3 PlcGetValid()

```
bool plc::DataValue::PlcGetValid( ) const [inline]
```

Get the valid flag and reset the dirty flag

Returns

`valid` True for valid data, False for invalid

Definition at line 375 of file plcBase.h.

References `GetValid()`, and `myuserdirty`.

Referenced by `plc::BaseRecord::PlcGetValid()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.4 PlcRead() [1/4]

```
template<typename T >
bool plc::DataValue::PlcRead (
    T & data ) const [inline]
```

Read data by the plc

Parameters

<i>data</i>	Data value reference (return)
-------------	-------------------------------

Definition at line 320 of file plcBase.h.

References myplcdirty, and Read().

Referenced by plc::BaseRecord::PlcRead().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.5 PlcRead() [2/4]

```
template<size_type N>
bool plc::DataValue::PlcRead (
    type_string_value(&) data[N] ) const [inline]
```

Read fixed length character array data by the plc

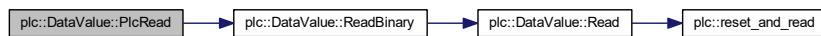
Parameters

<i>data</i>	Data value reference for a fixed length character array (return)
-------------	--

Definition at line 324 of file plcBase.h.

References myplcdirty, and ReadBinary().

Here is the call graph for this function:



8.14.3.6 PlcRead() [3/4]

```
bool plc::DataValue::PlcRead (
    type_string_value * data,
    size_type max ) const [inline]
```

Read character array (pchar) by the plc

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 329 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:



8.14.3.7 PlcRead() [4/4]

```
bool plc::DataValue::PlcRead (
    type_wstring_value * data,
    size_type max ) const [inline]
```

Read character array (pwchar) by the plc

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 334 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:



8.14.3.8 PlcReadBinary()

```
size_type plc::DataValue::PlcReadBinary (
    type_binary p,
    size_type len ) const [inline]
```

Read data as binary by the plc

Parameters

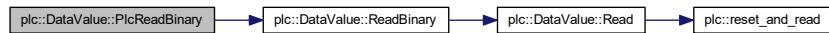
<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 358 of file plcBase.h.

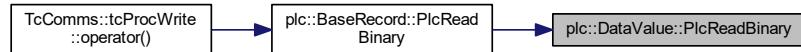
References myplcdirty, and ReadBinary().

Referenced by plc::BaseRecord::PlcReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.9 PlcSetValid()

```
void plc::DataValue::PlcSetValid (
    bool valid ) [inline]
```

Set the valid flag and set the dirty flag when flag changes

Parameters

<code>valid</code>	True for valid data, False for invalid
--------------------	--

Definition at line 372 of file plcBase.h.

References myplcdirty, and SetValid().

Referenced by plc::BaseRecord::PlcSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.10 PlcWrite() [1/4]

```
template<typename T >
bool plc::DataValue::PlcWrite (
    const T & data ) [inline]
```

Write data by the plc

Parameters

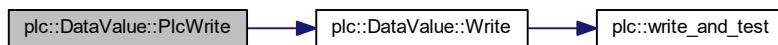
<i>data</i>	Data value reference
-------------	----------------------

Definition at line 338 of file plcBase.h.

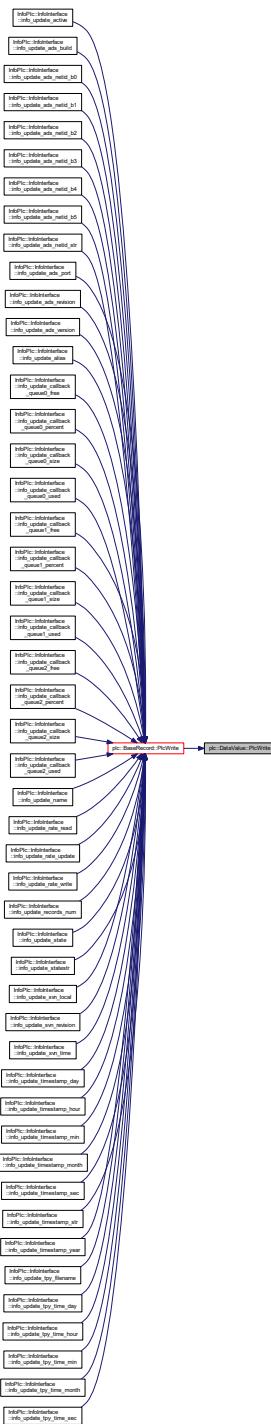
References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::PlcWrite().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.11 PlcWrite() [2/4]

```
template<size_type N>
bool plc::DataValue::PlcWrite (
    const type_string_value(&) data[N] ) [inline]
```

Write fixed length character array data by the plc

Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 342 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:



8.14.3.12 PlcWrite() [3/4]

```
bool plc::DataValue::PlcWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Write character array (pchar) by the plc

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 347 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



8.14.3.13 PlcWrite() [4/4]

```
bool plc::DataValue::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Write character array (wpchar) by the plc

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 352 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



8.14.3.14 PlcWriteBinary()

```
size_type plc::DataValue::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Write data as binary by the plc

Parameters

<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 363 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::PlcWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.15 Read() [1/5]

```
template<typename T >
bool plc::DataValue::Read (
    atomic_bool & dirty,
    T & data ) const [protected]
```

Read data

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

[DataValue::Read](#) (bool, Integral and floating point types)

Definition at line 87 of file plcBaseTemplate.h.

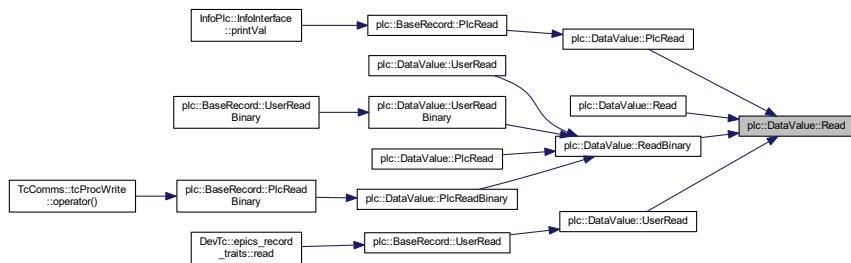
References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `mydata`, `mytype`, and `plc::reset_and_read()`.

Referenced by `PlcRead()`, `Read()`, `ReadBinary()`, and `UserRead()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.16 Read() [2/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_string & data ) const [protected]
```

Read string (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

Definition at line 287 of file plcBase.cpp.

References plc::dtString, mydata, mytype, and plc::reset_and_read().

Here is the call graph for this function:



8.14.3.17 Read() [3/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring & data ) const [protected]
```

Read wstring (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

Definition at line 299 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, mytype, and plc::reset_and_read().

Here is the call graph for this function:

**8.14.3.18 Read() [4/5]**

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_string_value * data,
    size_type max ) const [protected]
```

Read character array (pchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 314 of file plcBase.cpp.

References Read().

Here is the call graph for this function:



8.14.3.19 Read() [5/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring_value * data,
    size_type max ) const [protected]
```

Read character array (pwchar)

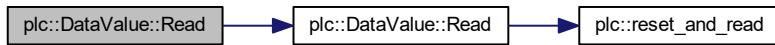
Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 328 of file plcBase.cpp.

References Read().

Here is the call graph for this function:



8.14.3.20 ReadBinary()

```
DataValue::size_type plc::DataValue::ReadBinary (
    atomic_bool & dirty,
    type_binary p,
    size_type len ) const [protected]
```

Read data as binary

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 398 of file plcBase.cpp.

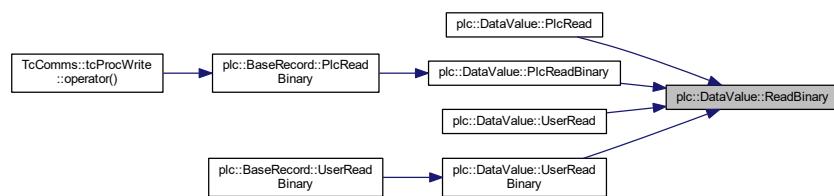
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, plc::DataValue<TypeDef::memory_order, mydata, mysize, mytype, and Read().

Referenced by PlcRead(), PlcReadBinary(), UserRead(), and UserReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.21 SetValid()

```

void plc::DataValue::SetValid (
    atomic_bool & dirty,
    bool valid ) [protected]
  
```

Set the valid flag and set the dirty flag when flag changes

Parameters

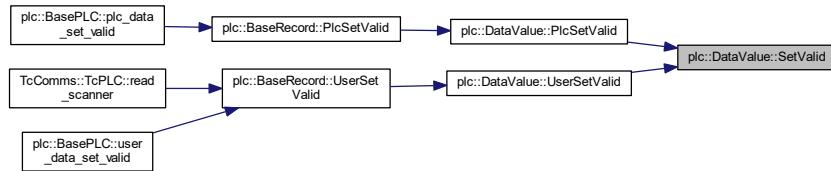
<i>dirty</i>	Reference to dirty flag (user or plc)
<i>valid</i>	True for valid data, False for invalid

Definition at line 494 of file plcBase.cpp.

References plc::DataValueTypeDef::memory_order, and myvalid.

Referenced by PlcSetValid(), and UserSetValid().

Here is the caller graph for this function:



8.14.3.22 UserGetValid()

```
bool plc::DataValue::UserGetValid ( ) const [inline]
```

Get the valid flag and reset the dirty flag

Returns

valid True for valid data, False for invalid

Definition at line 316 of file plcBase.h.

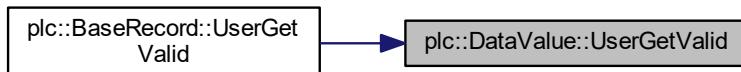
References GetValid(), and myplcdirty.

Referenced by plc::BaseRecord::UserGetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.23 UserRead() [1/4]

```
template<typename T >
bool plc::DataValue::UserRead (
    T & data ) const [inline]
```

Read data by the user

Parameters

<i>data</i>	Data value reference (return)
-------------	-------------------------------

Definition at line 260 of file plcBase.h.

References myuserdirty, and Read().

Referenced by plc::BaseRecord::UserRead().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.24 UserRead() [2/4]

```
template<size_type N>
bool plc::DataValue::UserRead (
    type_string_value(&) data[N] ) const [inline]
```

Read fixed length character array data by the user

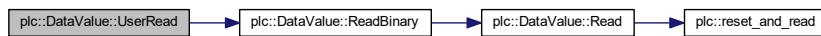
Parameters

<i>data</i>	Data value reference for a fixed length character array (return)
-------------	--

Definition at line 264 of file plcBase.h.

References myuserdirty, and ReadBinary().

Here is the call graph for this function:



8.14.3.25 UserRead() [3/4]

```
bool plc::DataValue::UserRead (
    type_string_value * data,
    size_type max ) const [inline]
```

Read character array (pchar) by the user

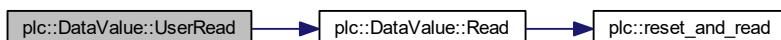
Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 269 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:



8.14.3.26 UserRead() [4/4]

```
bool plc::DataValue::UserRead (
    type_wstring_value * data,
    size_type max ) const [inline]
```

Read character array (pwchar) by the user

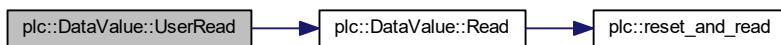
Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 274 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:

**8.14.3.27 UserReadBinary()**

```
size_type plc::DataValue::UserReadBinary (
    type_binary p,
    size_type len ) const [inline]
```

Read data as binary by the user

Parameters

<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 299 of file plcBase.h.

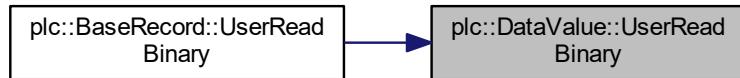
References myuserdirty, and ReadBinary().

Referenced by plc::BaseRecord::UserReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.28 UserSetValid()

```
void plc::DataValue::UserSetValid (
    bool valid ) [inline]
```

Set the valid flag and set the dirty flag when flag changes

Parameters

<code>valid</code>	True for valid data, False for invalid
--------------------	--

Definition at line 313 of file plcBase.h.

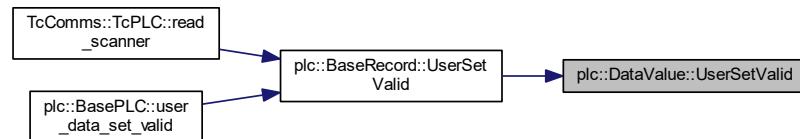
References myuserdirty, and SetValid().

Referenced by plc::BaseRecord::UserSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.29 UserWrite() [1/4]

```
template<typename T >
bool plc::DataValue::UserWrite (
    const T & data ) [inline]
```

Write data by the user

Parameters

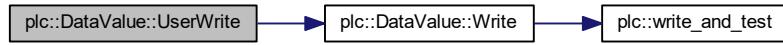
<i>data</i>	Data value reference
-------------	----------------------

Definition at line 278 of file plcBase.h.

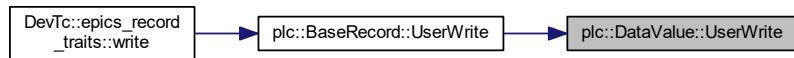
References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::UserWrite().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.30 UserWrite() [2/4]

```
template<size_type N>
bool plc::DataValue::UserWrite (
    const type_string_value(&) data[N] ) [inline]
```

Write fixed length character array data by the user

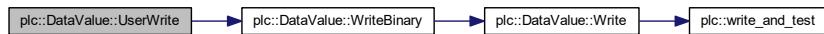
Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 282 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:



8.14.3.31 UserWrite() [3/4]

```
bool plc::DataValue::UserWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Write character array (pchar) by the user

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 288 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



8.14.3.32 UserWrite() [4/4]

```
bool plc::DataValue::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Write character array (wpchar) by the user

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 293 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:

**8.14.3.33 UserWriteBinary()**

```
size_type plc::DataValue::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Write data as binary by the user

Parameters

<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 304 of file plcBase.h.

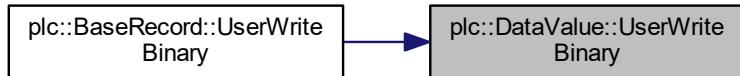
References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::UserWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.34 Write() [1/5]

```
template<typename T >
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const T & data ) [protected]
```

Write data

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference (return)

[DataValue::UserWrite](#) (bool, Integral and floating point types)

Definition at line 120 of file `plcBaseTemplate.h`.

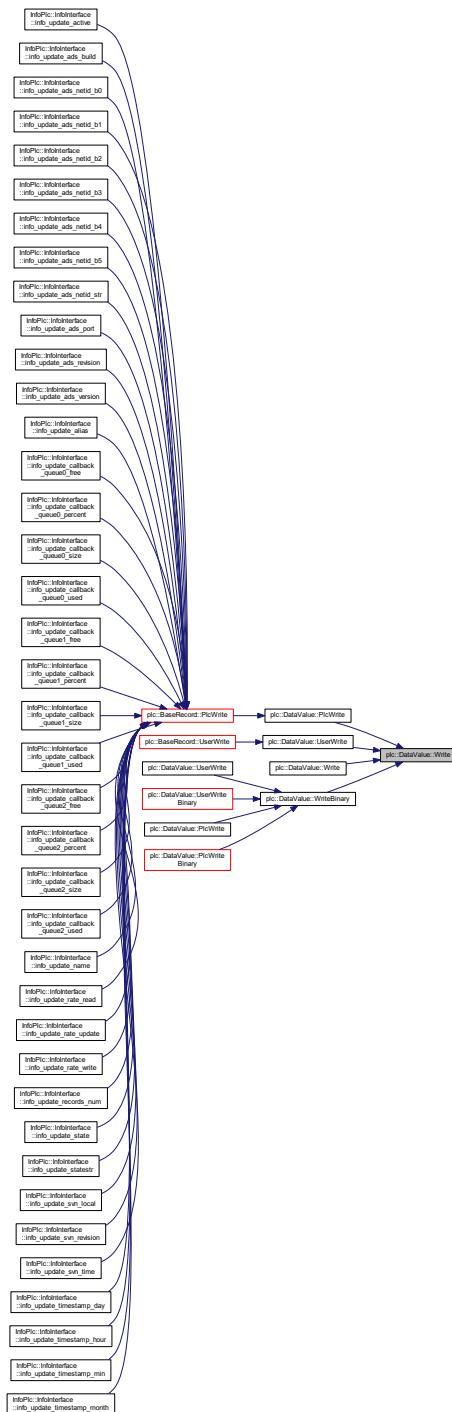
References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `mydata`, `mytype`, `myvalid`, and `plc::write_and_test()`.

Referenced by `PlcWrite()`, `UserWrite()`, `Write()`, and `WriteBinary()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.3.35 Write() [2 / 5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
```

```
const atomic_bool & pend,
const type_string & data ) [protected]
```

Write string (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference

Definition at line 342 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, mytype, myvalid, and plc::write_and_test().

Here is the call graph for this function:



8.14.3.36 Write() [3/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_wstring & data ) [protected]
```

Write wstring (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference

Definition at line 358 of file plcBase.cpp.

References plc::dtWString, mydata, mytype, myvalid, and plc::write_and_test().

Here is the call graph for this function:



8.14.3.37 Write() [4/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_string_value * data,
    size_type max ) [protected]
```

Write character array (pchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 371 of file plcBase.cpp.

References Write().

Here is the call graph for this function:



8.14.3.38 Write() [5/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
```

```
const atomic_bool & pend,
const type_wstring_value * data,
size_type max ) [protected]
```

Write character array (pwchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 384 of file plcBase.cpp.

References Write().

Here is the call graph for this function:

**8.14.3.39 WriteBinary()**

```
DataValue::size_type plc::DataValue::WriteBinary (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_binary p,
    size_type len ) [protected]
```

Write data as binary

Parameters

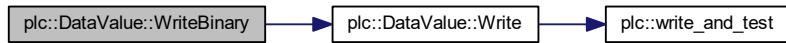
<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 446 of file plcBase.cpp.

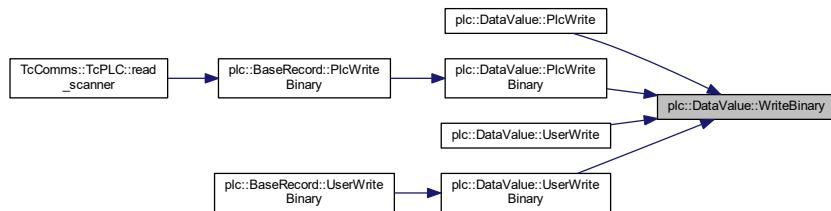
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, plc::DataValue<TypeDef>::memory_order, mydata, mysize, mytype, and Write().

Referenced by PlcWrite(), PlcWriteBinary(), UserWrite(), and UserWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.14.4 Member Data Documentation

8.14.4.1 mysize

`size_type` `plc::DataValue::mysize` [protected]

Size of allocated memory for simple types; size of string class for strings and size of data for binary

Definition at line 465 of file `plcBase.h`.

Referenced by `get_size()`, `Init()`, `IsValid()`, `operator=()`, `ReadBinary()`, and `WriteBinary()`.

The documentation for this class was generated from the following files:

- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.15 `plc::DataValueTraits< T >` Struct Template Reference

Data value traits.

```
#include <plcBase.h>
```

Public Types

- `typedef size_t size_type`
size type
- `typedef plc::data_type_enum data_type_enum`
enumerated type for data type
- `typedef T traits_type`
traits type
- `typedef std::atomic< T > traits_atomic`
atomic variable type

Public Member Functions

- `template<>`
`const DataValueTraits< DataValueTypeDef::type_bool >::data_type_enum data_enum`
Bool specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_int8 >::data_type_enum data_enum`
Int8 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_uint8 >::data_type_enum data_enum`
UInt8 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_int16 >::data_type_enum data_enum`
Int16 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_uint16 >::data_type_enum data_enum`
UInt16 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_int32 >::data_type_enum data_enum`
Int32 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_uint32 >::data_type_enum data_enum`
UInt32 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_float >::data_type_enum data_enum`
Float specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_double >::data_type_enum data_enum`
Double specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_string >::data_type_enum data_enum`
String specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_wstring >::data_type_enum data_enum`
WString specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_binary >::data_type_enum data_enum`
Binary specialization for DataValueTraits.

Static Public Attributes

- static const `data_type_enum data_enum = dtInvalid`
`data type enumertaion value`

8.15.1 Detailed Description

```
template<typename T>
struct plc::DataValueTraits< T >
```

Data value traits.

Traits class for data value

Definition at line 112 of file plcBase.h.

8.15.2 Member Data Documentation

8.15.2.1 data_enum

```
template<typename T>
const DataValueTraits< T >::data_type_enum plc::DataValueTraits< T >::data_enum = dtInvalid
[static]
```

data type enumertaion value

`DataValue::data_enum`

Definition at line 123 of file plcBase.h.

The documentation for this struct was generated from the following files:

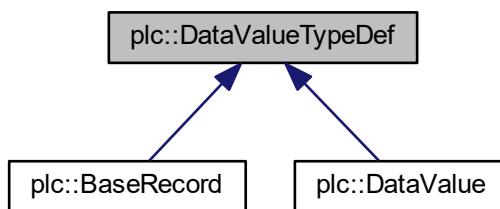
- `plcBase.h`
- `plcBaseTemplate.h`

8.16 plc::DataValueTypeDef Struct Reference

Collection of type definitions.

```
#include <plcBase.h>
```

Inheritance diagram for `plc::DataValueTypeDef`:



Public Types

- `typedef size_t size_type`
size type
- `typedef plc::data_type_enum data_type_enum`
enumerated type for data type
- `typedef bool type_bool`
bool type
- `typedef signed char type_int8`
1-byte integer type
- `typedef unsigned char type_uint8`
1-byte unsigned integer type
- `typedef short type_int16`
3-byte integer type
- `typedef unsigned short type_uint16`
3-byte unsigned integer type
- `typedef int type_int32`
4-byte integer type
- `typedef unsigned int type_uint32`
4-byte unsigned integer type
- `typedef long long type_int64`
8-byte integer type
- `typedef unsigned long long type_uint64`
8-byte unsigned integer type
- `typedef float type_float`
4-byte single precision floating point type
- `typedef double type_double`
4-byte double precision floating point type
- `typedef std::string type_string`
string type
- `typedef std::wstring type_wstring`
wstring type
- `typedef void * type_binary`
binary type
- `typedef std::string::value_type type_string_value`
string character type
- `typedef std::wstring::value_type type_wstring_value`
wstring character type
- `typedef DataValueTraits< type_bool >::traits_atomic atomic_bool`
atomic bool type
- `typedef DataValueTraits< type_int8 >::traits_atomic atomic_int8`
atomic 1-byte integer type
- `typedef DataValueTraits< type_uint8 >::traits_atomic atomic_uint8`
atomic 1-byte unsigned integer type
- `typedef DataValueTraits< type_int16 >::traits_atomic atomic_int16`
atomic 2-byte integer type
- `typedef DataValueTraits< type_uint16 >::traits_atomic atomic_uint16`
atomic 2-byte unsigned integer type
- `typedef DataValueTraits< type_int32 >::traits_atomic atomic_int32`
atomic 4-byte integer type
- `typedef DataValueTraits< type_uint32 >::traits_atomic atomic_uint32`

- `typedef DataValueTraits< type_int64 >::traits_atomic atomic_int64`
atomic 4-byte unsigned integer type
- `typedef DataValueTraits< type_uint64 >::traits_atomic atomic_uint64`
atomic 8-byte integer type
- `typedef DataValueTraits< type_float >::traits_atomic atomic_float`
atomic 4-byte single precision floating point type
- `typedef DataValueTraits< type_double >::traits_atomic atomic_double`
atomic 8-byte double precision floating point type
- `typedef DataValueTraits< type_string >::traits_atomic atomic_string`
atomic string type
- `typedef DataValueTraits< type_wstring >::traits_atomic atomic_wstring`
atomic wstring type
- `typedef DataValueTraits< type_binary >::traits_atomic atomic_binary`
atomic binary type
- `typedef DataValueTypeDef::type_uint64 time_type`
Define timestamp type.

Static Public Attributes

- `static const std::memory_order memory_order = std::memory_order_seq_cst`
memory order used for atomic access

8.16.1 Detailed Description

Collection of type definitions.

Type definitions for data value

Definition at line 129 of file plcBase.h.

The documentation for this struct was generated from the following file:

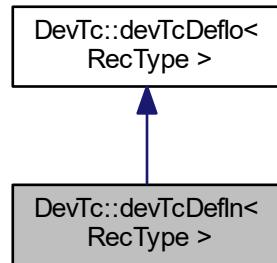
- `plcBase.h`

8.17 DevTc::devTcDefIn< RecType > Struct Template Reference

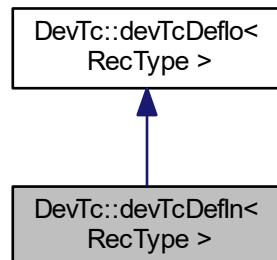
Device support input record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefIn< RecType >:



Collaboration diagram for DevTc::devTcDefIn< RecType >:



Public Member Functions

- [devTcDefIn \(\)](#)

Constructor.

Static Public Member Functions

- static long [init_read_record \(rec_type_ptr prec\)](#)

init callback for read records

- static long [read \(rec_type_ptr precord\)](#)

read callback

Additional Inherited Members

8.17.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefIn< RecType >
```

Device support input record.

Device Support Record for TwinCAT/ADS input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 347 of file devTc.h.

The documentation for this struct was generated from the following file:

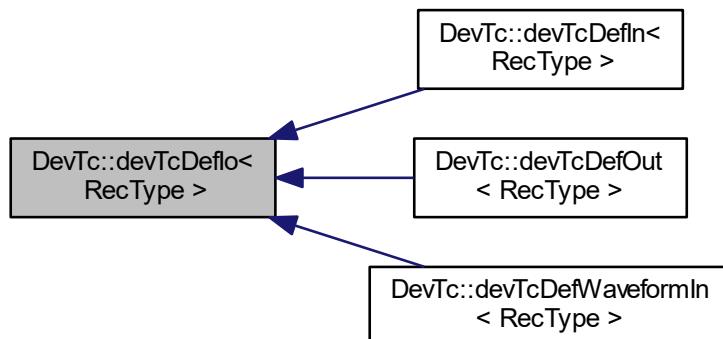
- [devTc.h](#)

8.18 DevTc::devTcDeflo< RecType > Struct Template Reference

Device support record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDeflo< RecType >:



Public Types

- **typedef epics_record_traits< RecType >::traits_type rec_type**
Record type: aiRecord, etc.
- **typedef rec_type * rec_type_ptr**
Pointer to record type.

Public Attributes

- long [number](#)
Number of support functions.
- DEVSUPFUN [report_fn](#)
Report support function.
- DEVSUPFUN [init_fn](#)
Init support function.
- DEVSUPFUN [init_record_fn](#)
Record init support function.
- DEVSUPFUN [get_ioint_info_fn](#)
IO/INT support function.
- DEVSUPFUN [io_fn](#)
Read/write support function.
- DEVSUPFUN [special_linconv_fn](#)
Linear conversion support function.

Protected Member Functions

- [devTcDeflo \(\)](#)
Hide constructor.

Static Protected Member Functions

- static long [get_ioint_info](#) (int cmd, dbCommon *prec, IOSCANPVT *ppvt)
IO/INT info callback.

8.18.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDeflo< RecType >
```

Device support record.

Deviced Support Record for generic TwinCAT/ADS IO This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 312 of file devTc.h.

The documentation for this struct was generated from the following file:

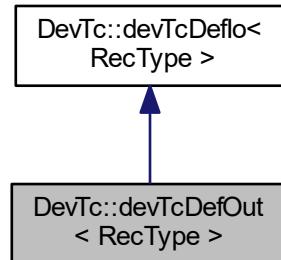
- [devTc.h](#)

8.19 DevTc::devTcDefOut< RecType > Struct Template Reference

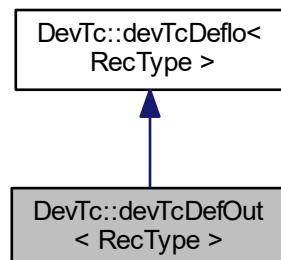
device support output record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefOut< RecType >:



Collaboration diagram for DevTc::devTcDefOut< RecType >:



Public Member Functions

- [devTcDefOut \(\)](#)

Constructor.

Static Public Member Functions

- static long [init_write_record \(rec_type_ptr prec\)](#)
init callback for write records
- static long [write \(rec_type_ptr precord\)](#)
write callback

Additional Inherited Members

8.19.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefOut< RecType >
```

device support output record.

Device Support Record for TwinCAT/ADS output This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 363 of file devTc.h.

The documentation for this struct was generated from the following file:

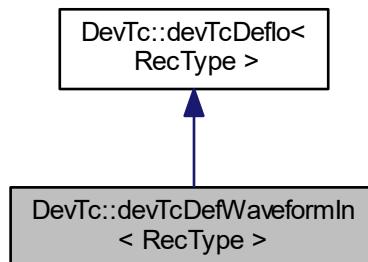
- [devTc.h](#)

8.20 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference

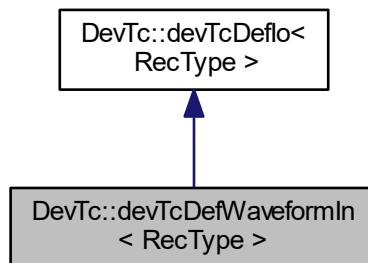
device support waveform record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefWaveformIn< RecType >:



Collaboration diagram for DevTc::devTcDefWaveformIn< RecType >:



Public Member Functions

- `devTcDefWaveformIn ()`

Constructor.

Static Public Member Functions

- static long `init_read_waveform_record (rec_type_ptr prec)`
init callback for read records
- static long `read_waveform (rec_type_ptr precord)`
read callback

Additional Inherited Members

8.20.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefWaveformIn< RecType >
```

device support waveform record.

Device Support Record for TwinCAT/ADS waveform input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 379 of file devTc.h.

The documentation for this struct was generated from the following file:

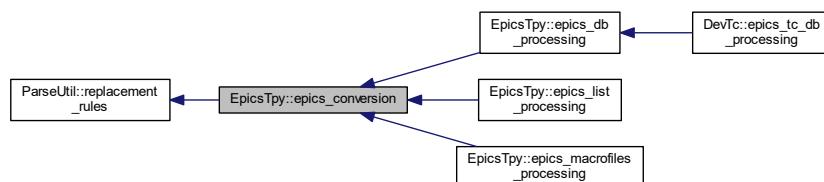
- `devTc.h`

8.21 EpicsTpy::epics_conversion Class Reference

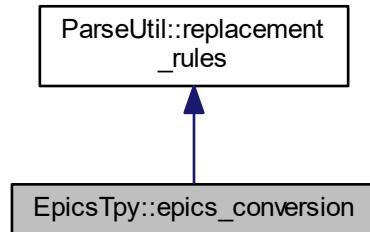
Epics conversion.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_conversion:



Collaboration diagram for EpicsTpy::epics_conversion:



Public Member Functions

- `epics_conversion ()`
Default constructor.
- `epics_conversion (case_type caseconv, bool noindex)`
- `epics_conversion (tc_epics_conv epics_conv, case_type caseconv, bool noldot, bool noindex)`
- `epics_conversion (int argc, const char *const argv[], bool argp[] = 0)`
- `int getopt (int argc, const char *const argv[], bool argp[] = 0)`
- `tc_epics_conv get_conversion_rule () const`
Get the conversion rule.
- `void set_conversion_rule (tc_epics_conv epics_conv)`
Set the conversion rule.
- `case_type get_case_rule () const`
Get the conversion rule.
- `void set_case_rule (case_type epics_conv)`
Set the conversion rule.
- `bool get_dot_rule () const`
Get the leadin dot rule.
- `void set_dot_rule (bool noldot)`
Set the leading dot rule.
- `bool get_array_rule () const`
Get the array index rule.
- `void set_array_rule (bool noindex)`
Set the array conversion rule.
- `std::string to_epics (const std::stringcase &name) const`

Protected Attributes

- `tc_epics_conv conv_rule`
Conversion rule.
- `case_type case_epics_names`
Case conversion rule.
- `bool no_leading_dot`
Leading dot conversion rule.
- `bool no_array_index`
Array index conversion rule.

Additional Inherited Members

8.21.1 Detailed Description

Epics conversion.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 59 of file TpyToEpics.h.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 epics_conversion() [1/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    case_type caseconv,
    bool noindex ) [inline]
```

Constructor

Parameters

<i>caseconv</i>	Case conversion specification
<i>noindex</i>	Eliminate array indices '[n]' with '_n'

Definition at line 68 of file TpyToEpics.h.

8.21.2.2 epics_conversion() [2/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    tc_epics_conv epics_conv,
    case_type caseconv,
    bool noldot,
    bool noindex ) [inline]
```

Constructor

Parameters

<i>epics_conv</i>	Epics conversion rule
<i>caseconv</i>	Case conversion specification
<i>noldot</i>	Eliminate leading dot in a name
<i>noindex</i>	Eliminate array indices '[n]' with '_n'

Definition at line 76 of file TpyToEpics.h.

8.21.2.3 epics_conversion() [3/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 87 of file TpyToEpics.h.

References getopt().

Here is the call graph for this function:



8.21.3 Member Function Documentation

8.21.3.1 getopt()

```
int EpicsTpy::epics_conversion::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/rn: Does not apply any special conversion rules /rd: Replaces dots with underscores in channel names /rl: LIGO standard conversion rule (default) /cp: Preserve case in EPICS channel names /cu: Force upper case in EPICS

channel names (default) /cl: Force lower case in EPICS channel names /nd: Eliminates leading dot in channel name (default) /yd: Leaves leading dot in channel name /ni: Replaces array brackets with underscore (default) /yi: Leave array indices as is

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

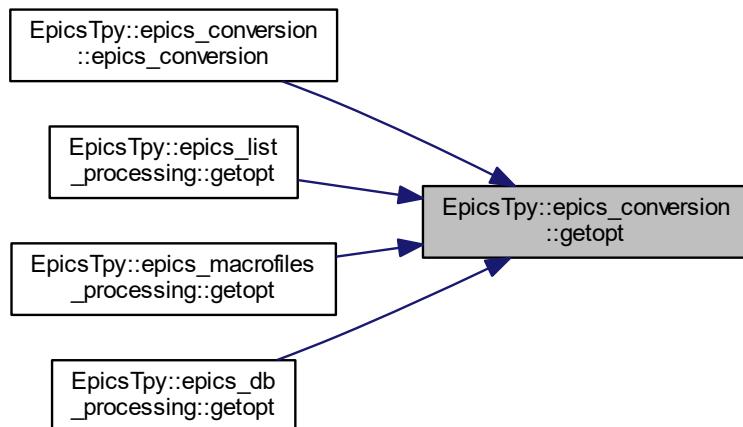
Number of arguments processed

Definition at line 25 of file TpyToEpics.cpp.

References `EpicsTpy::ligo_std`, `EpicsTpy::ligo_vac`, `EpicsTpy::lower_case`, `EpicsTpy::no_conversion`, `EpicsTpy::no_dot`, `EpicsTpy::preserve_case`, and `EpicsTpy::upper_case`.

Referenced by `epics_conversion()`, `EpicsTpy::epics_list_processing::getopt()`, `EpicsTpy::epics_macrofiles_processing::getopt()`, and `EpicsTpy::epics_db_processing::getopt()`.

Here is the caller graph for this function:



8.21.3.2 to_epics()

```
string EpicsTpy::epics_conversion::to_epics (
    const std::string& name ) const
```

Converts a TwinCAT or OPC name to an EPICS channel name

Parameters

<i>name</i>	TwinCAT/opc name
-------------	------------------

Returns

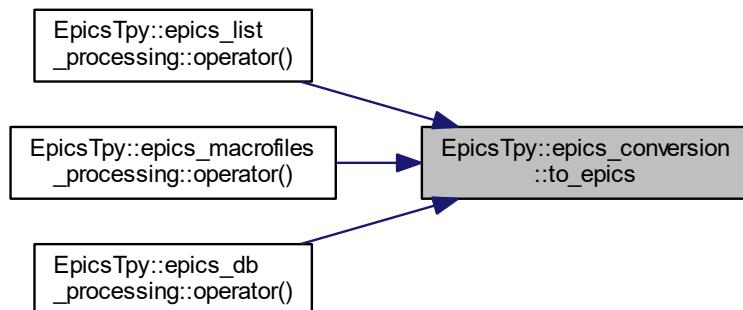
EPICS name

Definition at line 101 of file TpyToEpics.cpp.

References `EpicsTpy::ligo_std`, `EpicsTpy::ligo_vac`, `EpicsTpy::no_conversion`, `EpicsTpy::no_dot`, `EpicsTpy::preserve_case`, and `EpicsTpy::upper_case`.

Referenced by `EpicsTpy::epics_list_processing::operator()`, `EpicsTpy::epics_macrofiles_processing::operator()`, and `EpicsTpy::epics_db_processing::operator()`.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

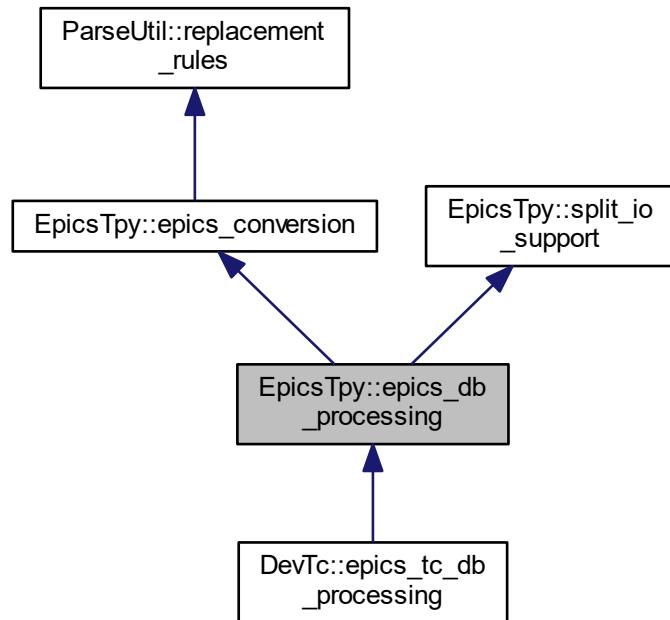
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.22 EpicsTpy::epics_db_processing Class Reference

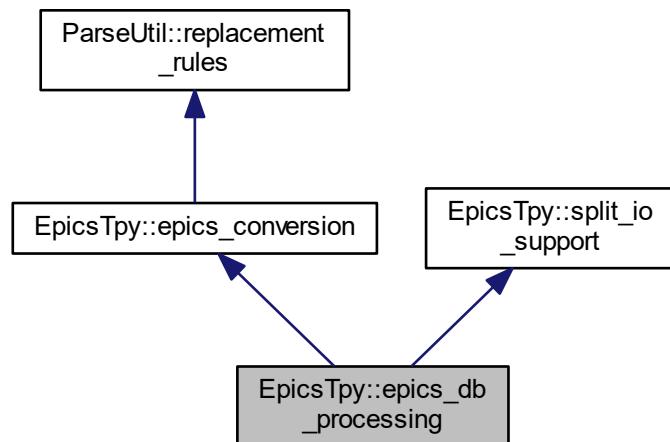
pics database record processing

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_db_processing:



Collaboration diagram for EpicsTpy::epics_db_processing:



Public Member Functions

- [epics_db_processing \(\)](#)

Default constructor.

- `epics_db_processing` (const `std::stringcase` &fname, int argc, const char *const argv[], bool argp[] = 0)
- int `getopt` (int argc, const char *const argv[], bool argp[] = 0)
- int `my getopt` (int argc, const char *const argv[], bool argp[] = 0)
- `device_support_type get_device_support () const`
 - Get device support conversion rule.*
- void `set_device_support (device_support_type devsup)`
 - Set device support conversion rule.*
- bool `operator() (const ParseUtil::process_arg &arg)`

Protected Member Functions

- bool `process_field_string (std::stringcase name, std::stringcase val)`
- bool `process_field_numeric (std::stringcase name, int val)`
- bool `process_field_numeric (std::stringcase name, double val)`
- bool `process_field_numeric (std::stringcase name, std::stringcase val)`
- bool `process_field_alarm (std::stringcase name, std::stringcase severity)`

Protected Attributes

- `device_support_type device_support`
 - Device support field conversion rule.*

Additional Inherited Members

8.22.1 Detailed Description

pics database record processing

Class for generatig an EPICS database record

Definition at line 694 of file TpyToEpics.h.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 `epics_db_processing()`

```
EpicsTpy::epics_db_processing::epics_db_processing (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with `epics_conversion::getopt`, `split_io_support::getopt` and `my getopt()`.

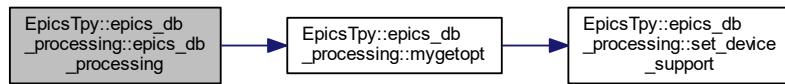
Parameters

<i>fname</i>	Output filename
<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 1183 of file TpyToEpics.cpp.

References [my getopt\(\)](#).

Here is the call graph for this function:



8.22.3 Member Function Documentation

8.22.3.1 getopt()

```
int EpicsTpy::epics_db_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line Processed options with [epics_conversion::getopt](#) and [my getopt](#).

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

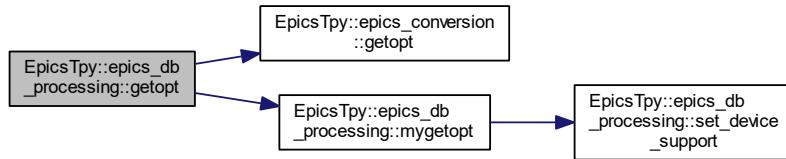
Returns

Number of arguments processed

Definition at line 1196 of file TpyToEpics.cpp.

References [EpicsTpy::epics_conversion::getopt\(\)](#), and [my getopt\(\)](#).

Here is the call graph for this function:



8.22.3.2 mygetopt()

```
int EpicsTpy::epics_db_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/devopc: Uses OPC name in INPUT/OUTPUT field (default) /devtc: Uses TwinCAT name in INPUT/OUTPUT fields instead of OPC

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

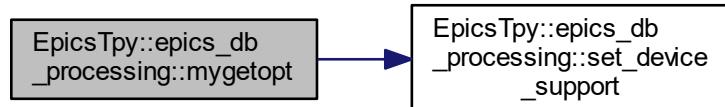
Number of arguments processed

Definition at line 1206 of file TpyToEpics.cpp.

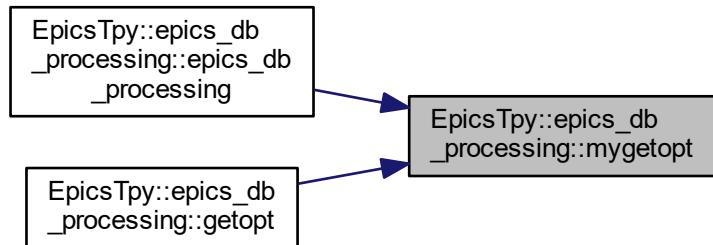
References `EpicsTpy::device_support_opc_name`, `EpicsTpy::device_support_tc_name`, and `set_device_support()`.

Referenced by `epics_db_processing()`, and `getopt()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.3.3 operator()()

```
bool EpicsTpy::epics_db_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

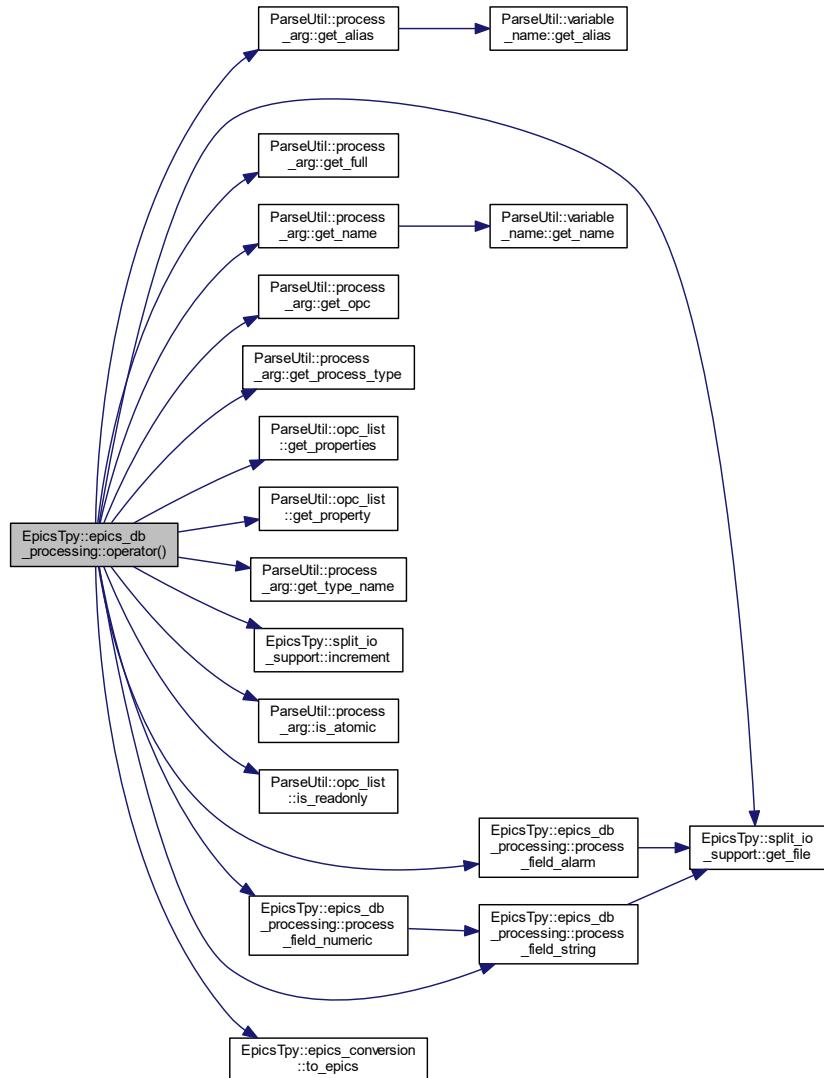
Returns

True if successfully processed

Definition at line 1239 of file TpyToEpics.cpp.

References device_support, EpicsTpy::device_support_opc_name, EpicsTpy::device_support_tc_name, EpicsTpy::EPICS_DB_COSV, EpicsTpy::EPICS_DB_DESC, EpicsTpy::EPICS_DB_DRVH, EpicsTpy::EPICS_DB_DRVL, EpicsTpy::EPICS_DB_DTYP, EpicsTpy::EPICS_DB_EGU, EpicsTpy::EPICS_DB_HHSV, EpicsTpy::EPI_CS_DB_HIGH, EpicsTpy::EPICS_DB_HIHI, EpicsTpy::EPICS_DB_HOPR, EpicsTpy::EPICS_DB_HSV, EpicsTpy::EPICS_DB_HYST, EpicsTpy::EPICS_DB_INP, EpicsTpy::EPICS_DB_LLSV, EpicsTpy::EPICS_DB_LOLO, EpicsTpy::EPICS_DB_LOPR, EpicsTpy::EPICS_DB_LOW, EpicsTpy::EPICS_DB_LSV, EpicsTpy::EPICS_DB_MAJOR, EpicsTpy::EPICS_DB_MINOR, EpicsTpy::EPICS_DB_ONAM, EpicsTpy::EPICS_DB_OSV, EpicsTpy::EPICS_DB_OUT, EpicsTpy::EPICS_DB_PINI, EpicsTpy::EPICS_DB_PREC, EpicsTpy::EPICS_DB_SCAN, EpicsTpy::EPICS_DB_TSE, EpicsTpy::EPICS_DB_UNSV, EpicsTpy::EPICS_DB_ZNAM, EpicsTpy::EPICS_DB_ZRST, EpicsTpy::EPICS_DB_ZRSV, EpicsTpy::EPICS_DB_ZRVL, EpicsTpy::EPICS_DB_ZSV, ParseUtil::process_arg::get_alias(), EpicsTpy::split_io_support::get_file(), ParseUtil::process_arg::get_full(), ParseUtil::process_arg::get_name(), ParseUtil::process_arg::get_opc(), ParseUtil::process_arg::get_process_type(), ParseUtil::opc_list::get_properties(), ParseUtil::opc_list::get_property(), ParseUtil::process_arg::get_type_name(), EpicsTpy::split_io_support::increment(), ParseUtil::process_arg::is_atomic(), ParseUtil::opc_list::is_READONLY(), EpicsTpy::MAX_EPICS_CHANNEL, EpicsTpy::MAX_EPICS_DESC, ParseUtil::OPC_PROP_ALIAS, ParseUtil::OPC_PROP_ALMC, ParseUtil::OPC_PROP_ALMDB, ParseUtil::OPC_PROP_ALMFFSV, ParseUtil::OPC_PROP_ALMH, ParseUtil::OPC_PROP_ALMH, ParseUtil::OPC_PROP_ALMHHSV, ParseUtil::OPC_PROP_ALMHHSV, ParseUtil::OPC_PROP_ALML, ParseUtil::OPC_PROP_ALMLL, ParseUtil::OPC_PROP_ALMLLSV, ParseUtil::OPC_PROP_ALMLSV, ParseUtil::OPC_PROP_ALMOSV, ParseUtil::OPC_PROP_ALMUNSV, ParseUtil::OPC_PROP_ALMZRSV, ParseUtil::OPC_PROP_ALMZSV, ParseUtil::OPC_PROP_CLOSE, ParseUtil::OPC_PROP_DESC, ParseUtil::OPC_PROP_DTYP, ParseUtil::OPC_PROP_FFST, ParseUtil::OPC_PROP_HIEU, ParseUtil::OPC_PROP_HIRANGE, ParseUtil::OPC_PROP_INOUT, ParseUtil::OPC_PROP_LOEU, ParseUtil::OPC_PROP_LORANGE, ParseUtil::OPC_PROP_OPEN, ParseUtil::OPC_PROP_PINI, ParseUtil::OPC_PROP_PLNAME, ParseUtil::OPC_PROP_PREC, ParseUtil::OPC_PROP_RECTYPE, ParseUtil::OPC_PROP_SERVER, ParseUtil::OPC_PROP_TSE, ParseUtil::OPC_PROP_UNIT, ParseUtil::OPC_PROP_ZRST, process_field_alarm(), process_field_numeric(), process_field_string(), ParseUtil::pt_bool, ParseUtil::pt_enum, ParseUtil::pt_int, ParseUtil::pt_real, ParseUtil::pt_string, and EpicsTpy::epics_conversion::to_epics().

Here is the call graph for this function:



8.22.3.4 process_field_alarm()

```
bool EpicsTpy::epics_db_processing::process_field_alarm (
    std::stringcase name,
    std::stringcase severity ) [protected]
```

Process a record field of type alarm

Parameters

<i>name</i>	Name of field
<i>severity</i>	Severity of alarm

Returns

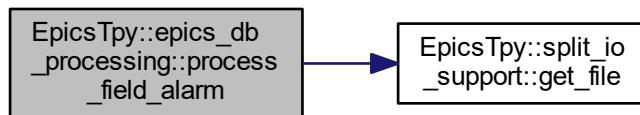
True if successful

Definition at line 1563 of file TpyToEpics.cpp.

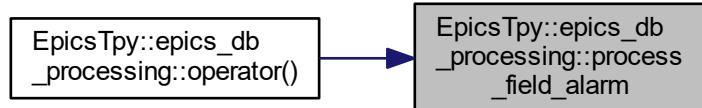
References `EpicsTpy::EPICS_DB_MAJOR`, `EpicsTpy::EPICS_DB_MINOR`, `EpicsTpy::EPICS_DB_NOALARM`, and `EpicsTpy::split_io_support::get_file()`.

Referenced by `operator()()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.3.5 process_field_numeric() [1/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    int val ) [protected]
```

Process a record field of numeric type

Parameters

<code>name</code>	Name of field
<code>val</code>	Value of field

Returns

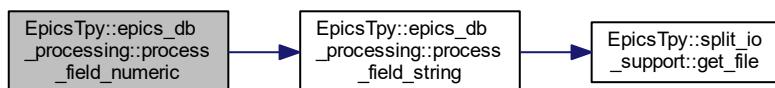
True if successful

Definition at line 1527 of file TpyToEpics.cpp.

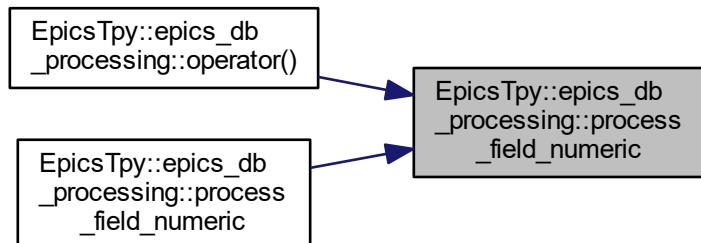
References process_field_string().

Referenced by operator()(), and process_field_numeric().

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.3.6 process_field_numeric() [2/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    double val )  [protected]
```

Process a record field of numeric type

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

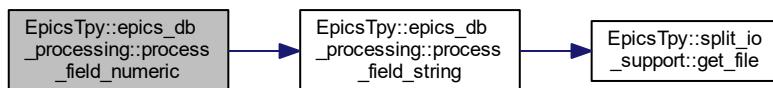
Returns

True if successful

Definition at line 1537 of file TpyToEpics.cpp.

References process_field_string().

Here is the call graph for this function:



8.22.3.7 process_field_numeric() [3/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    std::stringcase val ) [protected]
```

Process a record field of type string

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

Returns

True if successful

Definition at line 1547 of file TpyToEpics.cpp.

References process_field_numeric().

Here is the call graph for this function:



8.22.3.8 process_field_string()

```
bool EpicsTpy::epics_db_processing::process_field_string (
    std::stringcase name,
    std::stringcase val ) [protected]
```

Process a record field of type string

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

Returns

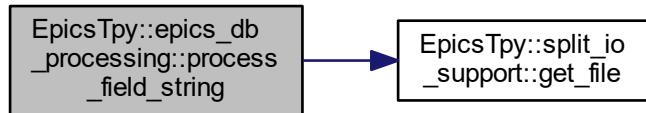
True if successful

Definition at line 1517 of file TpyToEpics.cpp.

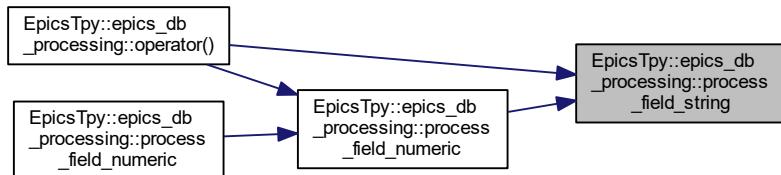
References EpicsTpy::split_io_support::get_file().

Referenced by operator()(), and process_field_numeric().

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

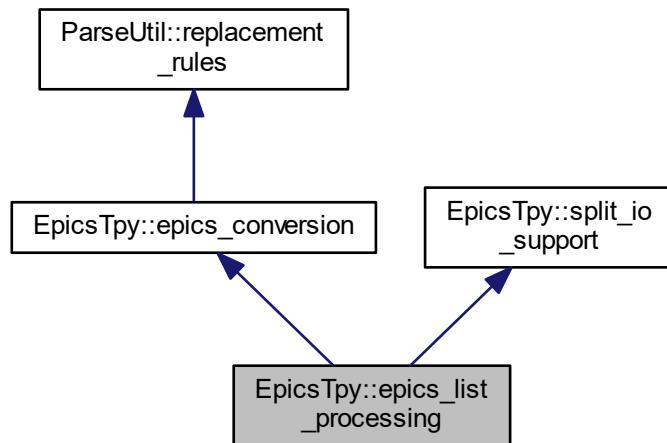
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.23 EpicsTpy::epics_list_processing Class Reference

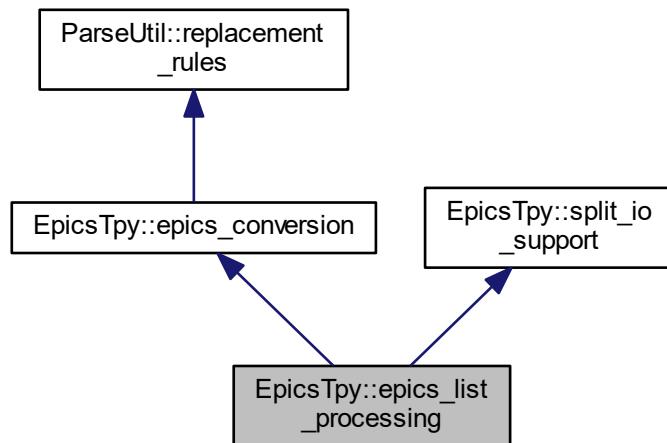
List processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_list_processing:



Collaboration diagram for EpicsTpy::epics_list_processing:



Public Member Functions

- [epics_list_processing \(\)](#)
Default constructor.
- [epics_list_processing \(listing_type ltype, bool ll=false\)](#)
- [epics_list_processing \(const std::string& fname, int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)
- [int getopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)
- [int mygetopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)
- [bool operator\(\) \(const ParseUtil::process_arg &arg\)](#)
- [listing_type get_listing \(\) const](#)
Get listing type.
- [void set_listing \(listing_type lt\)](#)
Set listing.
- [bool is_verbose \(\) const](#)
Is long listing?
- [void set_verbose \(bool vrbs\)](#)
Set long listing.

Protected Attributes

- [listing_type listing](#)
Listing type.
- [bool verbose](#)
long listing

Additional Inherited Members

8.23.1 Detailed Description

List processing.

Class for generating a channel list

Definition at line 427 of file TpyToEpics.h.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 epics_list_processing() [1/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    listing_type ltype,
    bool ll = false ) [inline], [explicit]
```

Constructor

Parameters

<i>ltype</i>	Type of listing
<i>l</i>	long listing

Definition at line 436 of file TpyToEpics.h.

8.23.2.2 epics_list_processing() [2/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    const std::string& fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with [epics_conversion::getopt](#), [split_io_support::getopt](#) and [my getopt\(\)](#).

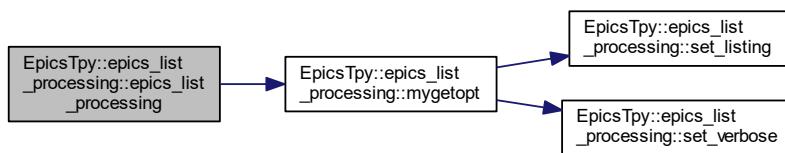
Parameters

<i>fname</i>	Output filename
<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 527 of file TpyToEpics.cpp.

References [my getopt\(\)](#).

Here is the call graph for this function:

**8.23.3 Member Function Documentation**

8.23.3.1 getopt()

```
int EpicsTpy::epics_list_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )
```

Parse a command line Processed options with [epics_conversion::getopt](#) and [mygetopt\(\)](#).

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

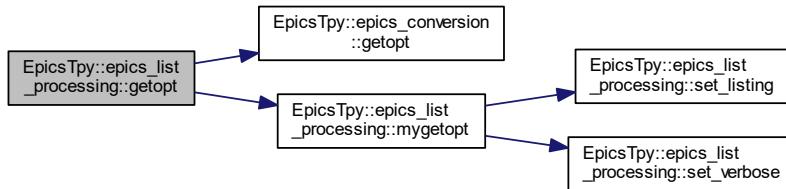
Returns

Number of arguments processed

Definition at line 540 of file TpyToEpics.cpp.

References EpicsTpy::epics_conversion::getopt(), and mygetopt().

Here is the call graph for this function:



8.23.3.2 mygetopt()

```
int EpicsTpy::epics_list_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/l: Generates a standard listing (default) /ll: Generates a long listing /lb: Generates an autoburn save/restore file /li: Generates a LIGO DAQ ini file

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Returns

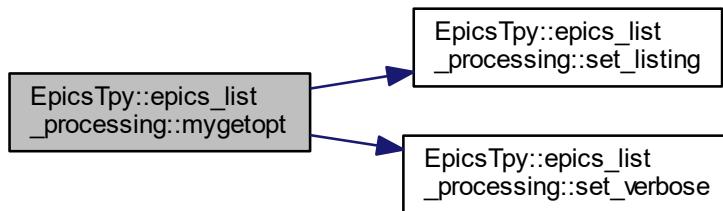
Number of arguments processed

Definition at line 550 of file TpyToEpics.cpp.

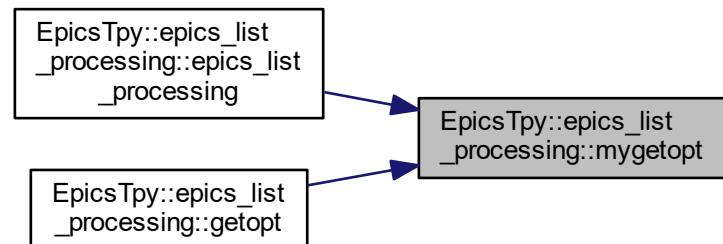
References `EpicsTpy::listing_autoburt`, `EpicsTpy::listing_daqini`, `EpicsTpy::listing_standard`, `set_listing()`, and `set_verbose()`.

Referenced by `epics_list_processing()`, and `getopt()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.3.3 operator()()

```
bool EpicsTpy::epics_list_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

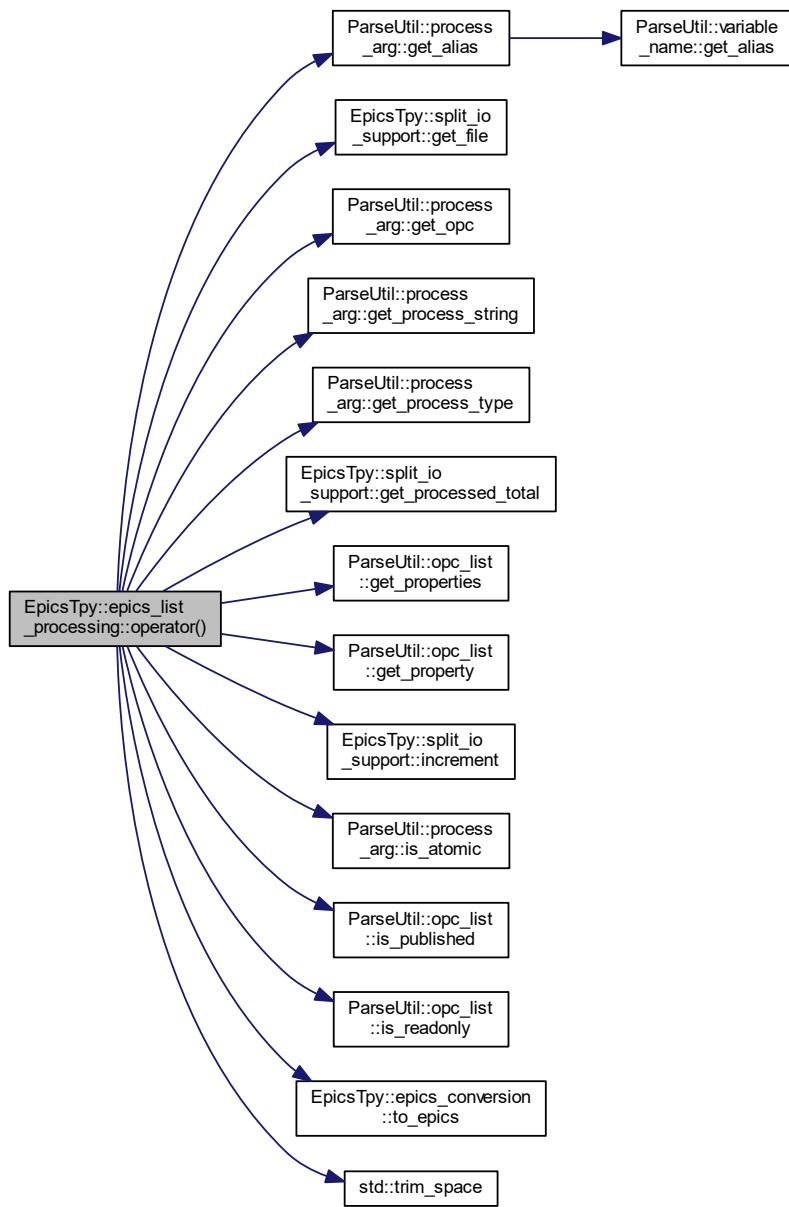
Returns

True if successfully processed

Definition at line 597 of file TpyToEpics.cpp.

References `ParseUtil::process_arg::get_alias()`, `EpicsTpy::split_io_support::get_file()`, `ParseUtil::process_arg::get_opc()`, `ParseUtil::process_arg::get_process_string()`, `ParseUtil::process_arg::get_process_type()`, `EpicsTpy::split_io_support::get_processed_total()`, `ParseUtil::opc_list::get_properties()`, `ParseUtil::opc_list::get_property()`, `EpicsTpy::split_io_support::increment()`, `ParseUtil::process_arg::is_atomic()`, `ParseUtil::opc_list::is_published()`, `ParseUtil::opc_list::is_READONLY()`, `EpicsTpy::LIGODAQ_DATATYPE_DEFAULT`, `EpicsTpy::LIGODAQ_DATATYPE_FLOAT`, `EpicsTpy::LIGODAQ_DATATYPE_INT32`, `EpicsTpy::LIGODAQ_DATATYPE_NAME`, `EpicsTpy::LIGODAQ_INI_HEADER`, `EpicsTpy::LIGODAQ_UNIT_DEFAULT`, `EpicsTpy::LIGODAQ_UNIT_NAME`, `EpicsTpy::LIGODAQ_UNIT_NONE`, `listing`, `EpicsTpy::listing_autoburt`, `EpicsTpy::listing_daqini`, `EpicsTpy::listing_standard`, `ParseUtil::OPC_PROP_CLOSE`, `ParseUtil::OPC_PROP_FFST`, `ParseUtil::OPC_PROP_OPEN`, `ParseUtil::OPC_PROP_UNIT`, `ParseUtil::OPC_PROP_ZRST`, `ParseUtil::pt_bool`, `ParseUtil::pt_enum`, `ParseUtil::pt_int`, `EpicsTpy::epics_conversion::to_epics()`, `std::trim_space()`, and `verbose`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

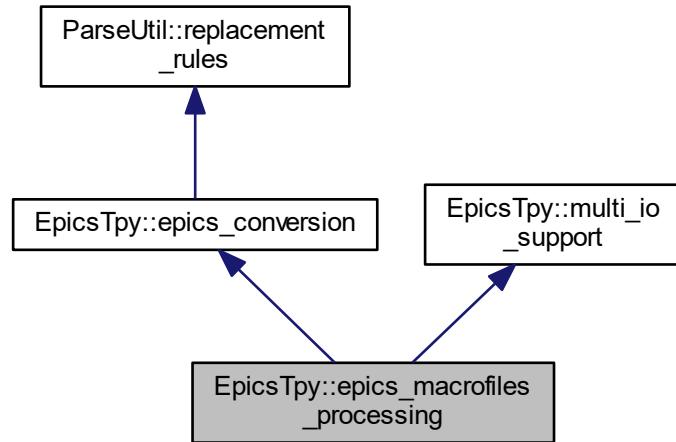
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.24 EpicsTpy::epics_macrofiles_processing Class Reference

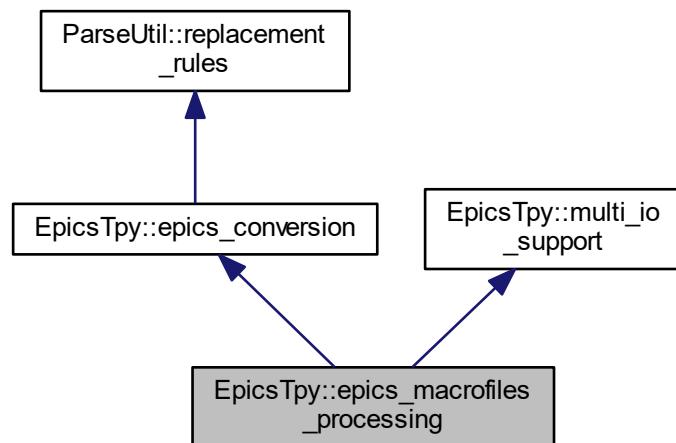
Macro file processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_macrofiles_processing:



Collaboration diagram for EpicsTpy::epics_macrofiles_processing:



Public Member Functions

- [epics_macrofiles_processing \(\)](#)
Default constructor.
- [epics_macrofiles_processing \(macrofile_type mt\)](#)

- `epics_macrofiles_processing` (const `std::stringcase` &pname, const `std::stringcase` &dbname, bool tcat3, int argc, const char *const argv[], bool argp[] = 0)
- `~epics_macrofiles_processing` ()
Destructor.
- `void flush` ()
flush all pending processing
- `int getopt` (int argc, const char *const argv[], bool argp[] = 0)
- `int mygetopt` (int argc, const char *const argv[], bool argp[] = 0)
- `bool operator()` (const `ParseUtil::process_arg` &arg)
- `macrofile_type get_macrofile_type` () const
Get listing type.
- `void set_macrofile_type` (macrofile_type m)
Set listing.
- `void set_plcname` (const `std::stringcase` &name)
Set PLC name.
- `const std::stringcase & get_plcname` () const
Get PLC name.
- `bool is_twincat3` () const
Is this twincat 3?
- `void set_twincat3` (bool tcat3 = true)
Set twincat 3 version?
- `std::stringcase to_filename` (const `std::stringcase` &epicsname)
Translate epics name to filename.
- `int get_processed_total` () const
Get number of processed channels.

Static Public Attributes

- static const `std::stringcase` errorstruct = "ErrorStruct"
Type name identifying an error struct ("ErrorStruct")
- static const `std::stringcase` errorlistext2 = "_Errors.exp"
TwinCAT 2.11: File extension identifying a list of error msgs ("_Errors.exp")
- static const `std::regex` errormatchregex2
TwinCAT 2.11: Regular expression to match the entire error record definition.
- static const `std::stringcase` errorlistext31 = "_Errors.TcGVL"
TwinCAT 3.1: File extension identifying a list of error msgs ("_Errors.TcGVL")
- static const `std::regex` errormatchregex31
TwinCAT 3.1: Regular expression to match the entire error record definition.
- static const `std::regex` errorsearchregex
TwinCAT 2.11/3.1: Regular expression to search for the actual error messages.

Protected Member Functions

- `bool process_record` (const `macro_record` &mrec, int level = 0)
Process top of stack.

Protected Attributes

- `macrofile_type macros`
Listing type.
- `std::stringcase plcname`
PLC name.
- `bool isTwinCAT3`
TwinCAT version.
- `macro_stack procstack`
Processing stack.
- `int rec_num`
Current number of processed channels (records)
- `filename_set missing`
set of missing input files

8.24.1 Detailed Description

Macro file processing.

Class for generating macro files to be used by medm

Definition at line 564 of file TpyToEpics.h.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 epics_macrofiles_processing() [1/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    macrofile_type mt )  [inline], [explicit]
```

Constructor

Parameters

<code>mt</code>	Type of macro
-----------------	---------------

Definition at line 584 of file TpyToEpics.h.

8.24.2.2 epics_macrofiles_processing() [2/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    const std::stringcase & pname,
    const std::stringcase & dname,
```

```

    bool tcat3,
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )

```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with [epics_conversion::getopt](#), [multi_io_support::getopt](#) and [my getopt\(\)](#).

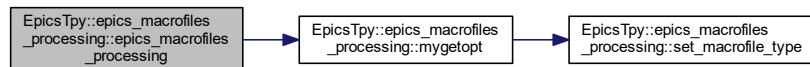
Parameters

<i>pname</i>	PLC name
<i>dname</i>	Directory name
<i>tcat3</i>	True if we are processing TwinCAT 3.1 files
<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 702 of file TpyToEpics.cpp.

References [EpicsTpy::all](#), and [my getopt\(\)](#).

Here is the call graph for this function:



8.24.3 Member Function Documentation

8.24.3.1 getopt()

```

int EpicsTpy::epics_macrofiles_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )

```

Parse a command line Processed options with [epics_conversion::getopt](#) and [my getopt\(\)](#).

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

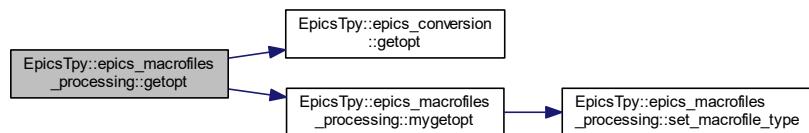
Returns

Number of arguments processed

Definition at line 727 of file TpyToEpics.cpp.

References EpicsTpy::epics_conversion::getopt(), and my getopt().

Here is the call graph for this function:

**8.24.3.2 my getopt()**

```
int EpicsTpy::epics_macrofiles_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/mf: Generate a macro file for each structure describing all fields /me: Generate a macro file for each structure describing the error messages /ma: Generate a macro file for each structure describing fields and errors (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

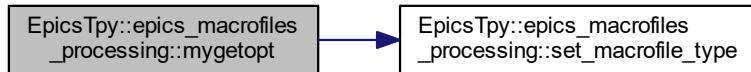
Number of arguments processed

Definition at line 737 of file TpyToEpics.cpp.

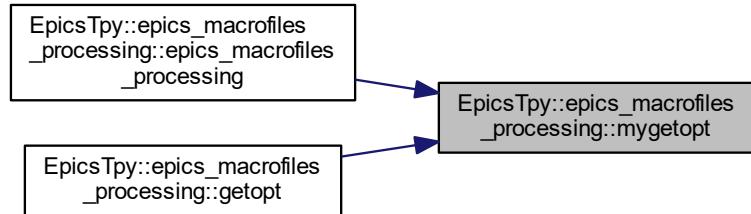
References EpicsTpy::all, EpicsTpy::errors, EpicsTpy::fields, and set_macrofile_type().

Referenced by epics_macrofiles_processing(), and getopt().

Here is the call graph for this function:



Here is the caller graph for this function:



8.24.3.3 operator()()

```
bool EpicsTpy::epics_macrofiles_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

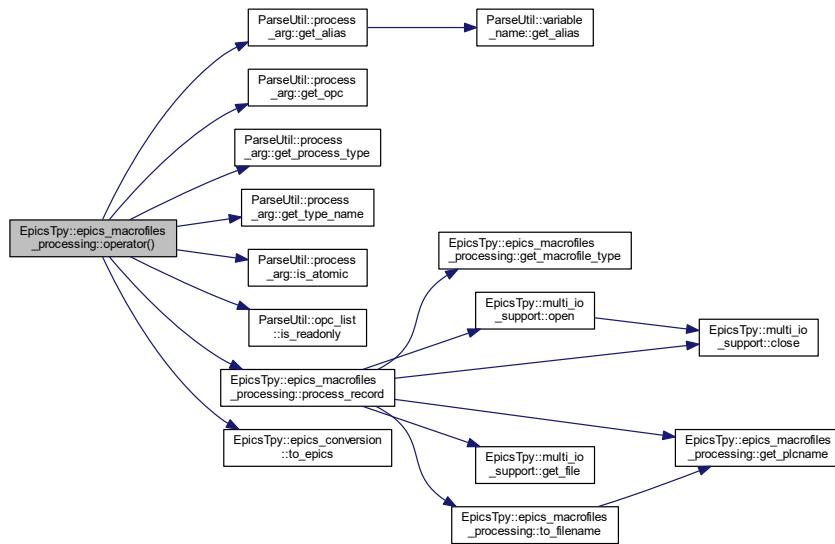
Returns

True if successfully processed

Definition at line 775 of file TpyToEpics.cpp.

References EpicsTpy::macro_record::back, errorstruct, ParseUtil::process_arg::get_alias(), ParseUtil::process_arg::get_opc(), ParseUtil::process_arg::get_process_type(), ParseUtil::process_arg::get_type_name(), ParseUtil::process_arg::is_atomic(), ParseUtil::opc_list::is_READONLY(), EpicsTpy::macro_record::iserror, EpicsTpy::macro_info::name, process_record(), procstack, ParseUtil::pt_INVALID, EpicsTpy::macro_info::ptype, EpicsTpy::macro_info::readonly, rec_num, EpicsTpy::macro_record::record, EpicsTpy::epics_conversion::to_epics(), and EpicsTpy::macro_info::type_n.

Here is the call graph for this function:



8.24.3.4 to_filename()

```
std::stringcase EpicsTpy::epics_macros_processing::to_filename (
    const std::stringcase & epicsname )
```

Translate epics name to filename.

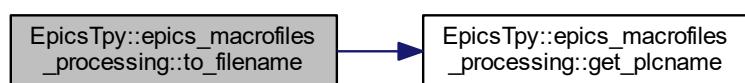
Translate epics name to filename

Definition at line 838 of file TpyToEpics.cpp.

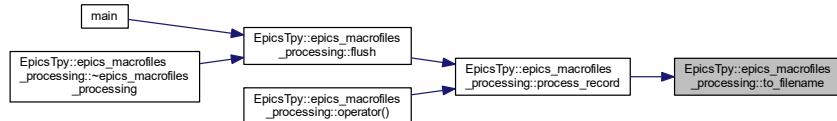
References `get_plcname()`, and `isTwinCAT3`.

Referenced by `process_record()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.25 DevTc::epics_record_traits< RecType > Struct Template Reference

Epics record traits.

```
#include <devTc.h>
```

Classes

- struct [traits_type](#)
Epics record type.

Public Types

- [typedef epicsFloat64 value_type](#)

Value type of (raw) value field.

Static Public Member Functions

- [static const char *const name \(\)](#)
Name of the record.
- [static value_type * val \(traits_type *prec\)](#)
Returns the (raw) value of a record.
- [static bool read \(traits_type *epicsrec, plc::BaseRecord *baserec\)](#)
Performs the read access on prec.
- [static bool write \(plc::BaseRecord *baserec, traits_type *epicsrec\)](#)
Performs the write access on prec.

Static Public Attributes

- static const aitEnum `value_ait_type` = aitEnumFloat64
- static const aitInt32 `value_count` = 0
- static const int `value_conversion` = 0
return value for read_io functions 0=default, 2=don't convert
- static const bool `input_record` = true
Indicates if this is an input record.
- static const bool `raw_record` = false
Indicates if this is a raw record.

8.25.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::epics_record_traits< RecType >
```

Epics record traits.

This traits class for Epics records.

Definition at line 273 of file devTc.h.

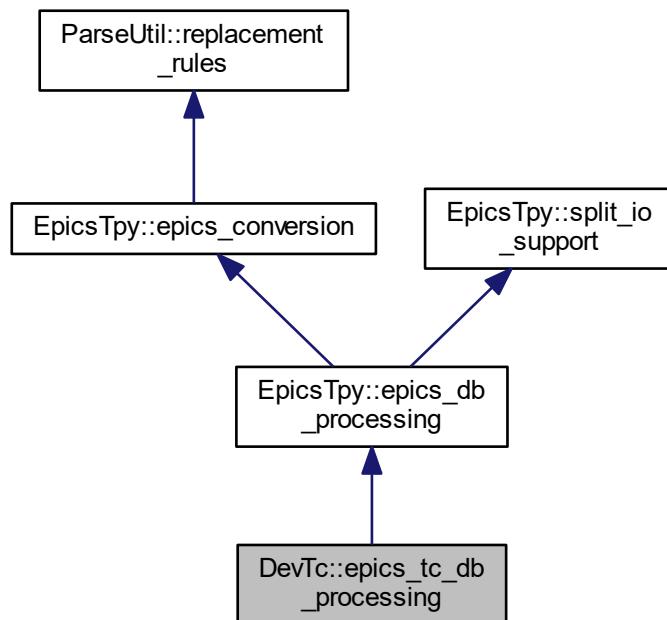
The documentation for this struct was generated from the following file:

- [devTc.h](#)

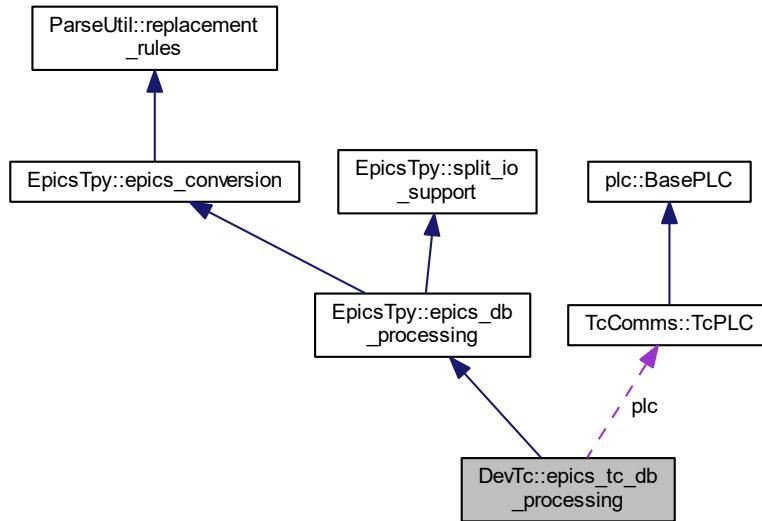
8.26 DevTc::epics_tc_db_processing Class Reference

EPICS/TCat db processing.

Inheritance diagram for DevTc::epics_tc_db_processing:



Collaboration diagram for DevTc::epics_tc_db_processing:



Public Member Functions

- `epics_tc_db_processing (TcComms::TcPLC &p, ParseUtil::replacement_table &rules, tc_listing_def *l=nullptr, tc_macro_def *m=nullptr)`
- `bool operator() (const ParseUtil::process_arg &arg)`
- `void flush ()`
Flush output files.
- `int get_invalid_records () const`
Get number of EPICS records without tc records.
- `bool patch_db_recordnames (std::stringcase &infodb)`

Protected Member Functions

- `epics_tc_db_processing (const epics_tc_db_processing &)`
Disable copy constructor.
- `epics_tc_db_processing & operator= (const epics_tc_db_processing &)`
Disable assignment operator.
- `void init_lists ()`
Init lists.
- `void done_lists ()`
Cleanup lists.
- `bool process_lists (const ParseUtil::process_arg &arg)`
- `bool process_list (filename_rule_list_tuple &listdef, const ParseUtil::process_arg &arg)`
- `void init_macros ()`
Init macros.
- `void done_macros ()`
Cleanup macros.
- `bool process_macros (const ParseUtil::process_arg &arg)`
- `bool process_macro (dirname_arg_macro_tuple ¯odef, const ParseUtil::process_arg &arg)`

Protected Attributes

- `TcComms::TcPLC * plc`
`Pointer to PLC class.`
- `tc_listing_def * lists`
`Pointer to a set of listings.`
- `tc_macro_def * macros`
`Pointer to macros.`
- `int invnum`
`Number of EPICS records without tc records.`

Additional Inherited Members

8.26.1 Detailed Description

EPICS/TCat db processing.

Class for generating an EPICS database and tc record

Definition at line 99 of file drvTc.cpp.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 epics_tc_db_processing()

```
DevTc::epics_tc_db_processing::epics_tc_db_processing (
    TcComms::TcPLC & p,
    ParseUtil::replacement_table & rules,
    tc_listing_def * l = nullptr,
    tc_macro_def * m = nullptr ) [inline], [explicit]
```

Default constructor

Parameters

<code>p</code>	PLC
<code>rules</code>	Replacement rules
<code>l</code>	Pointer to list definitions
<code>m</code>	Pointer to macro definition

Definition at line 106 of file drvTc.cpp.

References EpicsTpy::device_support_tc_name.

8.26.3 Member Function Documentation

8.26.3.1 operator()()

```
bool DevTc::epics_tc_db_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successful

8.26.3.2 patch_db_recordnames()

```
bool DevTc::epics_tc_db_processing::patch_db_recordnames (
    std::stringcase & infodb )
```

Patch channel names in info database

Parameters

<i>infodb</i>	EPICS database
---------------	----------------

Returns

True if no errors

8.26.3.3 process_list()

```
bool DevTc::epics_tc_db_processing::process_list (
    filename_rule_list_tuple & listdef,
    const ParseUtil::process_arg & arg ) [protected]
```

Process a listing

Parameters

<i>listdef</i>	filename/rule pair defining a listing
<i>arg</i>	Process argument describign the variable and type

Returns

True if successful

8.26.3.4 process_lists()

```
bool DevTc::epics_tc_db_processing::process_lists (
    const ParseUtil::process_arg & arg ) [protected]
```

Process all listings

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successful

8.26.3.5 process_macro()

```
bool DevTc::epics_tc_db_processing::process_macro (
    dirname_arg_macro_tuple & macrodef,
    const ParseUtil::process_arg & arg ) [protected]
```

Process a macro

Parameters

<i>macrodef</i>	filename/rule pair defining a macro
<i>arg</i>	Process argument describign the variable and type

Returns

True if successful

8.26.3.6 process_macros()

```
bool DevTc::epics_tc_db_processing::process_macros (
    const ParseUtil::process_arg & arg ) [protected]
```

Process all macros

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

Returns

True if successful

The documentation for this class was generated from the following file:

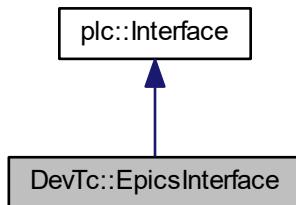
- [drvTc.cpp](#)

8.27 DevTc::EpicsInterface Class Reference

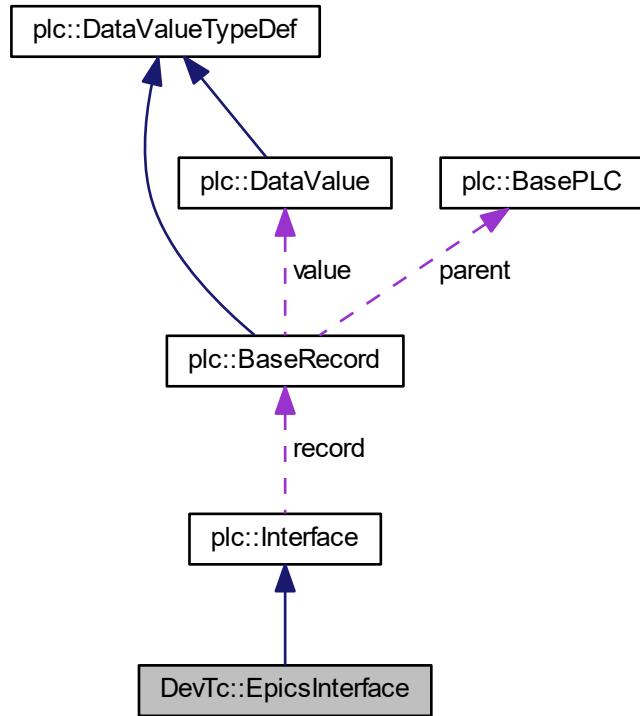
Epics interface class.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::EpicsInterface:



Collaboration diagram for DevTc::EpicsInterface:



Public Member Functions

- `EpicsInterface (plc::BaseRecord &dval)`
Constructor.
- `~EpicsInterface ()`
Deconstructor.
- `void set_isPassive (bool passive)`
Set isPassive.
- `bool get_isCallback () const`
Get isCallback.
- `void set_isCallback (bool isCb)`
Set isCallback.
- `void set_pEpicsRecord (dbCommon *pEpRecord)`
Set pEpicsRecord.
- `bool get_callbackRequestPending () const`
Get callbackRequestPending.
- `const CALLBACK & callback () const`
Get pointer to callback structure.
- `CALLBACK & callback ()`
Get pointer to callback structure.
- `const IOSCANPVT & ioscan () const`

- `IOSCANPVT & ioscan ()`

Get reference to io scan list pointer.
- `IOSCANPVT get_ioscan () const`

Get pointer to io scan list.
- `void set_ioscan (const IOSCANPVT ioscan)`

Set pointer to io scan list.
- `virtual bool push () override`

Makes a call to the EPICS dbProcess function.
- `virtual bool pull () override`

Does nothing.

Static Public Member Functions

- `static int get_callback_queue_size (int pri)`
- `static int get_callback_queue_used (int pri)`
- `static int get_callback_queue_free (int pri)`

Protected Member Functions

- `void ioscan_reset (int bitnum)`

Reset ioscan use flag.

Protected Attributes

- `bool isPassive`
- `bool isCallback`
- `dbCommon * pEpicsRecord`

Pointer to the EPICS record.
- `std::mutex ioscanmux`

IOSCAN mutex.
- `IOSCANPVT ioscanpvt`

Pointer to IO scan list.
- `std::atomic< unsigned int > ioscan_inuse`

Scan in progress (bit encoded value from priorities)
- `CALLBACK callbackval`

Callback structure.

Friends

- `void complete_io_scan (EpicsInterface *, IOSCANPVT, int)`

8.27.1 Detailed Description

Epics interface class.

This is a class for an EPICS Interface

Definition at line 86 of file devTc.h.

The documentation for this class was generated from the following files:

- [devTc.h](#)
- [devTc.cpp](#)

8.28 std::std::hash< std::stringcase > Struct Template Reference

hash for case insensitive string.

```
#include <stringcase_hash.h>
```

Public Member Functions

- `std::size_t operator()(const stringcase &str) const`

8.28.1 Detailed Description

```
template<>
struct std::std::hash< std::stringcase >
```

hash for case insensitive string.

This is a function specialization for case sensitive strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a string.

Definition at line 12 of file stringcase_hash.h.

8.28.2 Member Function Documentation

8.28.2.1 operator()()

```
std::size_t std::std::hash< std::stringcase >::operator() (
    const stringcase & str ) const
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv_32_str and fnv_64_str <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

Parameters

<code>str</code>	string to hash
------------------	----------------

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

- [stringcase_hash.h](#)

8.29 std::std::hash< std::wstringcase > Struct Template Reference

hash for case insensitive unicode string.

```
#include <stringcase_hash.h>
```

Public Member Functions

- `std::size_t operator()(const wstringcase &str) const`

8.29.1 Detailed Description

```
template<>
struct std::std::hash< std::wstringcase >
```

hash for case insensitive unicode string.

This is a function specialization for case sensitive unicode strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a unicode string.

Definition at line 28 of file `stringcase_hash.h`.

8.29.2 Member Function Documentation

8.29.2.1 operator()()

```
std::size_t std::std::hash< std::wstringcase >::operator() (
    const wstringcase & str ) const
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from `fnv_32_str` <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

Parameters

<code>str</code>	string to hash
------------------	----------------

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

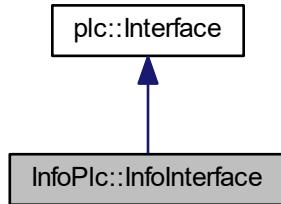
- [stringcase_hash.h](#)

8.30 InfoPlc::InfoInterface Class Reference

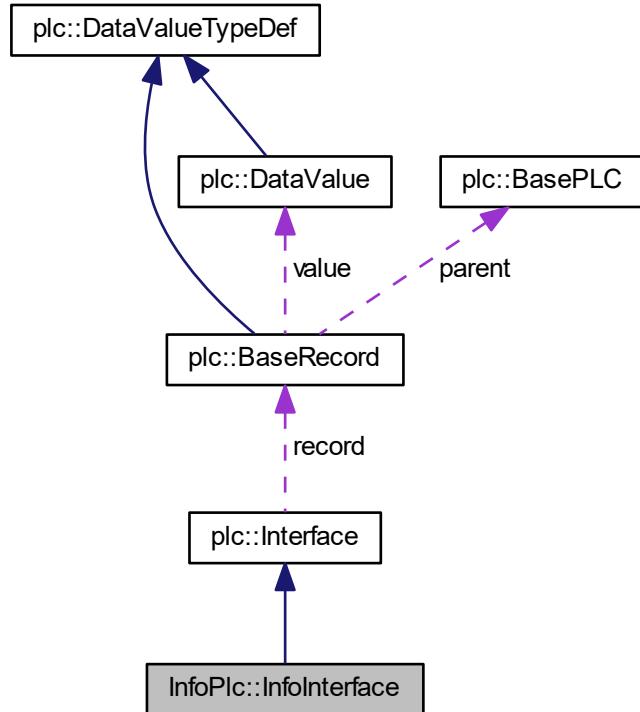
Info interface.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::InfoInterface:



Collaboration diagram for InfoPlc::InfoInterface:



Public Member Functions

- `InfoInterface (plc::BaseRecord &dval)`
- `InfoInterface (plc::BaseRecord &dval, const std::stringcase &id, const std::stringcase &name, const std::stringcase &type)`
- `~InfoInterface ()`
Deconstructor.
- `virtual bool push () override`
push data
- `virtual bool pull () override`
pull data
- `virtual bool update ()`
write data
- `virtual void printVal (FILE *fp)`
- `virtual const char * get_symbol_name () const`

Static Public Member Functions

- `template<class Function >`
- `static int get_infodb (const std::stringcase &prefix, const std::stringcase &plcaddr, Function &proc)`

Protected Member Functions

- bool `info_update_name ()`
info update: Name of PLC
- bool `info_update_alias ()`
info update: Alias name
- bool `info_update_active ()`
info update: Running state of PLC
- bool `info_update_state ()`
info update: AMS state of PLC
- bool `info_update_statestr ()`
info update: AMS state of PLC
- bool `info_update_timestamp_str ()`
info update: PLC time stamp
- bool `info_update_timestamp_year ()`
info update: Year of PLC time stamp
- bool `info_update_timestamp_month ()`
info update: Month of PLC time stamp
- bool `info_update_timestamp_day ()`
info update: Day of PLC time stamp
- bool `info_update_timestamp_hour ()`
info update: Hour of PLC time stamp
- bool `info_update_timestamp_min ()`
info update: Minute of PLC time stamp
- bool `info_update_timestamp_sec ()`
info update: Second of PLC time stamp
- bool `info_update_rate_read ()`
info update: Period of read scanner in ms
- bool `info_update_rate_write ()`
info update: Period of write scanner in ms
- bool `info_update_rate_update ()`
info update: Period of update scanner in ms
- bool `info_update_records_num ()`
info update: Number of EPICS records
- bool `info_update_tpy_filename ()`
info update: Name of tpy file
- bool `info_update_tpy_valid ()`
info update: Validity of tpy file
- bool `info_update_tpy_time_str ()`
info update: Modification time of tpy file
- bool `info_update_tpy_time_year ()`
info update: Year of tpy file time
- bool `info_update_tpy_time_month ()`
info update: Month of tpy file time
- bool `info_update_tpy_time_day ()`
info update: Day of tpy file time
- bool `info_update_tpy_time_hour ()`
info update: Hour of tpy file time
- bool `info_update_tpy_time_min ()`
info update: Minute of tpy file time
- bool `info_update_tpy_time_sec ()`

- `bool info_update_ads_version ()`
info update: ADS library version
- `bool info_update_ads_revision ()`
info update: ADS library revision
- `bool info_update_ads_build ()`
info update: ADS library build
- `bool info_update_ads_port ()`
info update: ADS/AMS port of PLC
- `bool info_update_ads_netid_str ()`
info update: ADS/AMS address of PLC
- `bool info_update_ads_netid_b0 ()`
info update: ADS/AMS address b0
- `bool info_update_ads_netid_b1 ()`
info update: ADS/AMS address b1
- `bool info_update_ads_netid_b2 ()`
info update: ADS/AMS address b2
- `bool info_update_ads_netid_b3 ()`
info update: ADS/AMS address b3
- `bool info_update_ads_netid_b4 ()`
info update: ADS/AMS address b4
- `bool info_update_ads_netid_b5 ()`
info update: ADS/AMS address b5
- `bool info_update_svn_local ()`
info update: SVN local modifications
- `bool info_update_svn_revision ()`
info update: SVN revision, if fully committed
- `bool info_update_svn_time ()`
info update: SVN compile time
- `bool info_update_callback_queue0_size ()`
info update: Size of low priority callback queue
- `bool info_update_callback_queue0_used ()`
info update: Used entries in low priority callback queue
- `bool info_update_callback_queue0_free ()`
info update: Free entries low priority callback queue
- `bool info_update_callback_queue0_percent ()`
info update: Usage percentage of low priority callback queue
- `bool info_update_callback_queue1_size ()`
info update: Size of medium priority callback queue
- `bool info_update_callback_queue1_used ()`
info update: Used entries in medium priority callback queue
- `bool info_update_callback_queue1_free ()`
info update: Free entries medium priority callback queue
- `bool info_update_callback_queue1_percent ()`
info update: Usage percentage of medium priority callback queue
- `bool info_update_callback_queue2_size ()`
info update: Size of high priority callback queue
- `bool info_update_callback_queue2_used ()`
info update: Used entries in high priority callback queue
- `bool info_update_callback_queue2_free ()`
info update: Free entries high priority callback queue
- `bool info_update_callback_queue2_percent ()`
info update: Usage percentage of high priority callback queue

Protected Attributes

- `std::stringcase tCatName`
Name of TCat symbol.
- `std::stringcase tCatType`
Data type in TCat.
- `update_enum update_freq`
Update frequency.
- `info_update_method info_update`
pointer to info update method

Static Protected Attributes

- `static const info_dbrecord_list dbinfo_list`
List of db info records.

8.30.1 Detailed Description

Info interface.

This is a class for a Info interface

Definition at line 55 of file infoPlc.h.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 InfoInterface() [1/2]

```
InfoPlc::InfoInterface::InfoInterface (
    plc::BaseRecord & dval ) [inline], [explicit]
```

Constructor

Parameters

<code>dval</code>	BaseRecord that this interface is part of
-------------------	---

Definition at line 60 of file infoPlc.h.

8.30.2.2 InfoInterface() [2/2]

```
InfoPlc::InfoInterface::InfoInterface (
    plc::BaseRecord & dval,
```

```
const std::stringcase & id,
const std::stringcase & name,
const std::stringcase & type )
```

Constructor

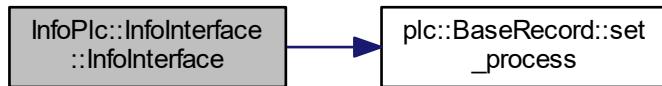
Parameters

<i>dval</i>	BaseRecord that this interface is part of
<i>id</i>	Short name info symbol
<i>name</i>	Full name of TCat symbol
<i>type</i>	Name of TCat data type

Definition at line 514 of file infoPlc.cpp.

References dbinfo_list, InfoPlc::done, info_update, InfoPlc::once, ParseUtil::pt_binary, ParseUtil::pt_enum, plc::Interface::record, plc::BaseRecord::set_process(), tCatType, and update_freq.

Here is the call graph for this function:



8.30.3 Member Function Documentation

8.30.3.1 get_infodb()

```
template<class Function >
int InfoPlc::InfoInterface::get_infodb (
    const std::stringcase & prefix,
    const std::stringcase & plcaddr,
    Function & proc ) [static]
```

Processes all info records

Parameters

<i>prefix</i>	Prefix to channel names in the info database
<i>plcaddr</i>	PLC associated with the info record
<i>proc</i>	database processing class

Returns

Returns Number of info records processed

Definition at line 12 of file infoPlcTemplate.h.

References dbinfo_list, and ParseUtil::publish.

8.30.3.2 get_symbol_name()

```
virtual const char* InfoPlc::InfoInterface::get_symbol_name ( ) const [inline], [virtual]
```

Get symbol name

Returns

Symbol name associated with this interface

Reimplemented from [plc::Interface](#).

Definition at line 95 of file infoPlc.h.

References tCatName.

8.30.3.3 printVal()

```
void InfoPlc::InfoInterface::printVal (
    FILE * fp ) [virtual]
```

Prints TCat symbol value and information

Parameters

<i>fp</i>	File to print symbol to
-----------	-------------------------

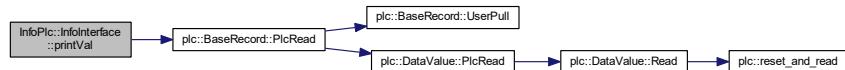
This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file fp.

Reimplemented from [plc::Interface](#).

Definition at line 563 of file infoPlc.cpp.

References plc::BaseRecord::PlcRead(), plc::Interface::record, tCatName, and tCatType.

Here is the call graph for this function:



8.30.4 Member Data Documentation

8.30.4.1 dbinfo_list

```
const info\_dbrecord\_list InfoPlc::InfoInterface::dbinfo_list [static], [protected]
```

List of db info records.

List of db info tuples.

Definition at line 212 of file infoPlc.h.

Referenced by `get_infodb()`, and `InfoInterface()`.

The documentation for this class was generated from the following files:

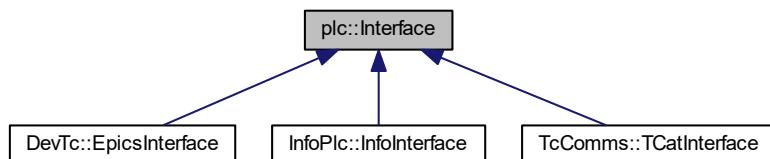
- [infoPlc.h](#)
- [infoPlc.cpp](#)
- [infoPlcTemplate.h](#)

8.31 plc::Interface Class Reference

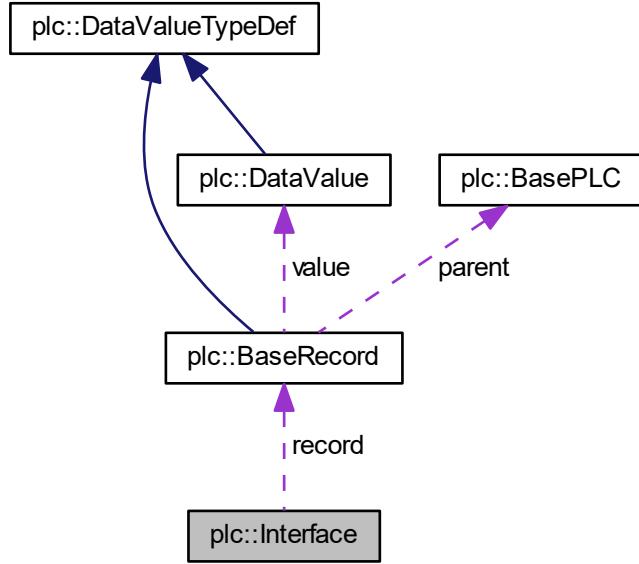
Abstract interface.

```
#include <plcBase.h>
```

Inheritance diagram for plc::Interface:



Collaboration diagram for plc::Interface:



Public Member Functions

- `Interface (BaseRecord &dval)`
- `virtual ~Interface ()`
Destructor.
- `BaseRecord & get_record ()`
Return a pointer to the tag/channel record.
- `const BaseRecord & get_record () const`
Return a pointer to the tag/channel record.
- `virtual bool push ()=0`
Pure virtual method indicating that the value needs to be pushed.
- `virtual bool pull ()=0`
Pure virtual method indicating that the value needs to be pulled.
- `BasePLC * get_parent ()`
Get parent PLC that owns this record.
- `const BasePLC * get_parent () const`
Get parent PLC that owns this record.
- `virtual void printVal (FILE *fp)`
- `virtual const char * get_symbol_name () const`
Get symbol name.

Protected Attributes

- `BaseRecord & record`
Pointer to tag/channel record associated with this interface.

8.31.1 Detailed Description

Abstract interface.

This is a base class for an abstract interface to access the PLC (slave) or the user side (master).

Definition at line 33 of file plcBase.h.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 Interface()

```
plc::Interface::Interface (
    BaseRecord & dval ) [inline], [explicit]
```

Constructor

Parameters

<i>dval</i>	Reference to a tag/channel record
-------------	-----------------------------------

Definition at line 38 of file plcBase.h.

8.31.3 Member Function Documentation

8.31.3.1 printVal()

```
virtual void plc::Interface::printVal (
    FILE * fp ) [inline], [virtual]
```

Print values to file

Parameters

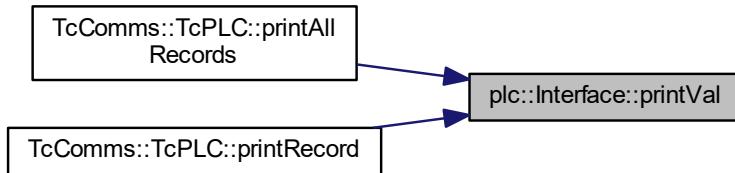
<i>fp</i>	File pointer
-----------	--------------

Reimplemented in [TcComms::TCatInterface](#), and [InfoPlc::InfoInterface](#).

Definition at line 58 of file plcBase.h.

Referenced by [TcComms::TcPLC::printAllRecords\(\)](#), and [TcComms::TcPLC::printRecord\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

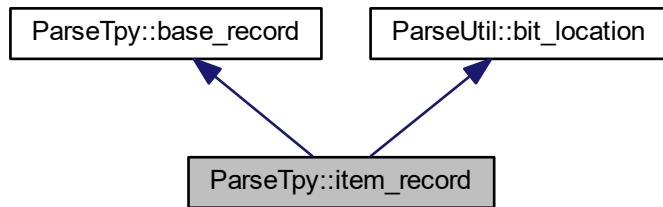
- [plcBase.h](#)
- [plcBase.cpp](#)

8.32 ParseTpy::item_record Class Reference

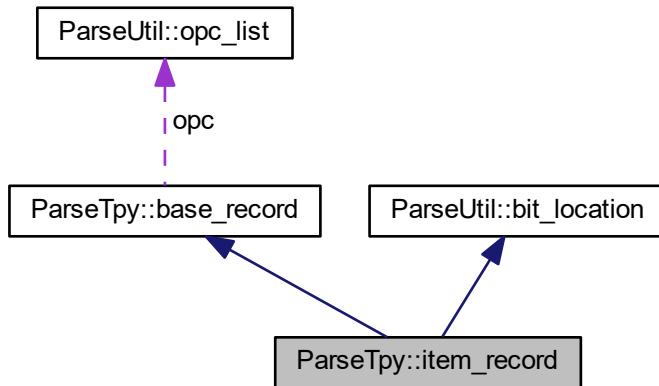
item record

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::item_record:



Collaboration diagram for ParseTpy::item_record:



Public Member Functions

- [item_record \(\)](#)

Default constructor.

Additional Inherited Members

8.32.1 Detailed Description

item record

This class stores typed items.

Definition at line 238 of file `ParseTpy.h`.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.33 EpicsTpy::macro_info Struct Reference

Macro information.

```
#include <TpyToEpics.h>
```

Public Member Functions

- [macro_info \(\)](#)
Default constructor.

Public Attributes

- [ParseUtil::process_type_enum ptype](#)
Process Type.
- [std::stringcase name](#)
name of type
- [std::stringcase type_n](#)
type definition
- [bool readonly](#)
readonly

8.33.1 Detailed Description

Macro information.

This structure describes a field

Definition at line 516 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

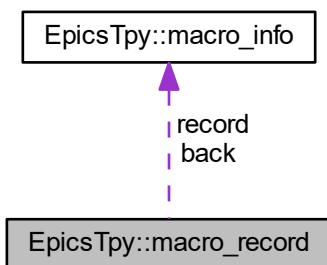
- [TpyToEpics.h](#)

8.34 EpicsTpy::macro_record Struct Reference

Macro record.

```
#include <TpyToEpics.h>
```

Collaboration diagram for EpicsTpy::macro_record:



Public Attributes

- [macro_info record](#)
name of structure
- [bool iserror](#)
is an ErrorStruct
- [bool haserror](#)
contains an ErrorStruct
- [int erroridx](#)
index if fields list to ErrorStruct
- [macro_list fields](#)
List of fields.
- [macro_info back](#)
name of upper level structure

8.34.1 Detailed Description

Macro record.

This structure describes a record/struct

Definition at line 537 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

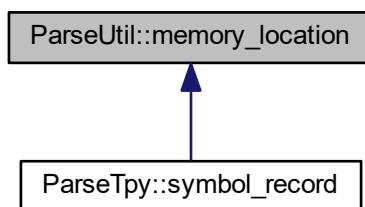
- [TpyToEpics.h](#)

8.35 ParseUtil::memory_location Class Reference

Memory location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::memory_location:



Public Member Functions

- `memory_location ()`
Default constructor.
- `memory_location (int ig, int io, int bs)`
- `memory_location (const std::stringcase &s)`
- `bool isValid () const`
Validity.
- `int get_igroup () const`
Get IGroup.
- `void set_igroup (int ig)`
Set IGroup.
- `int get_ioffset () const`
Get IOffset.
- `void set_ioffset (int io)`
Set IOffset.
- `int get_bytesize () const`
Get BitSize.
- `void set_bytesize (int bs)`
Set BitSize.
- `bool set_section (const bit_location &loc)`
- `std::stringcase get () const`
- `bool set (const std::stringcase &s)`

Protected Attributes

- `int igrup`
Memory group.
- `int ioffset`
Memory offset.
- `int bytesize`
Memory size in bits.

8.35.1 Detailed Description

Memory location.

This structure holds a memory location

Definition at line 263 of file ParseUtil.h.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 `memory_location()` [1/2]

```
ParseUtil::memory_location::memory_location (
    int ig,
    int io,
    int bs )  [inline]
```

Constructor

Parameters

<i>ig</i>	Index group
<i>io</i>	Index offset
<i>bs</i>	Size in bytes

Definition at line 272 of file ParseUtil.h.

8.35.2.2 memory_location() [2/2]

```
ParseUtil::memory_location::memory_location (
    const std::stringcase & s ) [inline]
```

Constructor

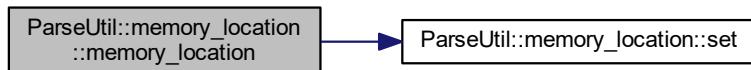
Parameters

<i>s</i>	Definition string
----------	-------------------

Definition at line 276 of file ParseUtil.h.

References set().

Here is the call graph for this function:



8.35.3 Member Function Documentation

8.35.3.1 get()

```
std::stringcase ParseUtil::memory_location::get ( ) const
```

Gets a string representation of a memory location

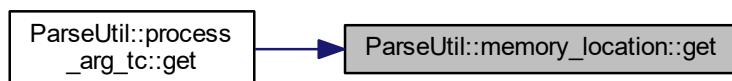
Returns

string with format "igroup/offset:size", empty on error

Definition at line 268 of file ParseUtil.cpp.

Referenced by ParseUtil::process_arg_tc::get().

Here is the caller graph for this function:



8.35.3.2 set()

```
bool ParseUtil::memory_location::set (
    const std::string& s )
```

Set the memory location using a string of the form: "igroup/offset:size" where the size is in bytes

Parameters

<code>s</code>	String describing memory location
----------------	-----------------------------------

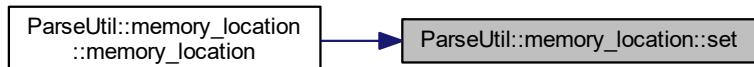
Returns

True if successful

Definition at line 281 of file ParseUtil.cpp.

Referenced by memory_location().

Here is the caller graph for this function:



8.35.3.3 `set_section()`

```
bool ParseUtil::memory_location::set_section (
    const bit_location & loc )
```

Set a sub section

Parameters

<i>loc</i>	Bit location within memory region
------------	-----------------------------------

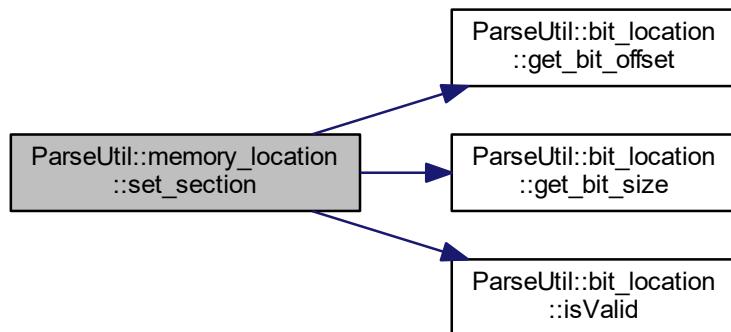
Returns

True if section within bounds, false otherwise

Definition at line 253 of file ParseUtil.cpp.

References `ParseUtil::bit_location::get_bit_offset()`, `ParseUtil::bit_location::get_bit_size()`, and `ParseUtil::bit_location::isValid()`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

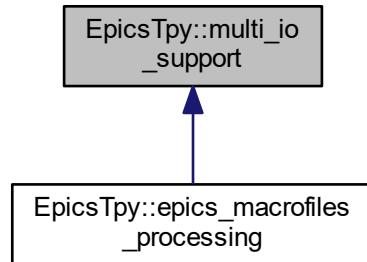
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.36 EpicsTpy::multi_io_support Class Reference

Multiple IO support.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::multi_io_support:



Public Member Functions

- `multi_io_support ()`
Default constructor.
- `multi_io_support (const std::stringcase &dbname)`
Constructor.
- `multi_io_support (const std::stringcase &dbname, int argc, const char *const argv[], bool argp[] = 0)`
- `~multi_io_support ()`
Destructor.
- `bool operator! () const`
Return error.
- `bool open (const std::stringcase &fname, const std::stringcase &io = "w", bool superrmsg = false)`
Open file for reading/writing.
- `void close ()`
Close file.
- `FILE * get_file () const`
Get file handle.
- `void set_outdirname (const std::stringcase &dbname)`
Set output directory name.
- `const std::stringcase & get_outdirname () const`
Get output directory name.
- `void set_in dirname (const std::stringcase &dbname)`
Set input directory name.
- `const std::stringcase & get_in dirname () const`
Get input directory name.
- `const std::stringcase & get_filename () const`
Get full filename.
- `io_filestat fileread () const`
Reading.
- `int get_filein_total () const`
Get number of read files.
- `int get_fileout_total () const`
Get number of written files.

Protected Member Functions

- void `set_filename` (const `std::stringcase` &fname)
Set output filename.
- int `getopt` (int argc, const char *const argv[], bool argp[] = 0)

Protected Attributes

- `std::stringcase outdirname`
Directory name.
- `std::stringcase indirname`
Directory name.
- `std::stringcase filename`
Current filename.
- `io_filestat filestat`
reading or writing?
- FILE * `filehandle`
Output file.
- int `file_num_in`
Current file number of processed read only channels (records)
- int `file_num_out`
Current file number of processed input/output channels (records)

8.36.1 Detailed Description

Multiple IO support.

Multi file IO support Supports a directory argument and opens files within

Definition at line 313 of file TpyToEpics.h.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 multi_io_support()

```
EpicsTpy::multi_io_support::multi_io_support (
    const std::stringcase & dname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

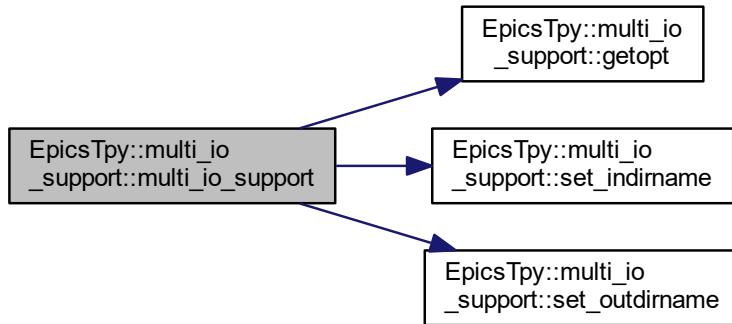
Parameters

<code>dname</code>	Name of output directory
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 332 of file TpyToEpics.h.

References getopt(), set_indirname(), and set_outdirname().

Here is the call graph for this function:



8.36.3 Member Function Documentation

8.36.3.1 getopt()

```
int EpicsTpy::multi_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline], [protected]
```

Parse a command line The format is the same as the arguments passed to the main program `argv[0]` is program name and will be ignored. The `argp` boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least `argc` long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. `getopt` will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be <code>argc</code>

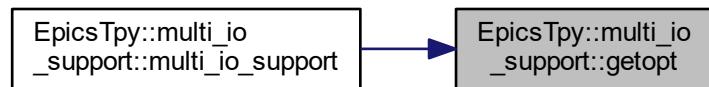
Returns

Number of arguments processed

Definition at line 391 of file TpyToEpics.h.

Referenced by multi_io_support().

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.37 ParseUtil::opc_list Class Reference

OPC list.

```
#include <ParseUtil.h>
```

Public Member Functions

- [`opc_list \(\)`](#)
Default constructor.
- [`opc_list \(opc_enum state, const property_map &map\)`](#)
Constructor.
- [`opc_enum get_opc_state \(\) const`](#)
Get opc state.
- [`void set_opc_state \(opc_enum state\)`](#)
Set opc state.
- [`const property_map & get_properties \(\) const`](#)
Get opc property list.
- [`property_map & get_properties \(\)`](#)
Get opc property list.
- [`const property_map::const_iterator get_property \(int key\) const`](#)
Get the specified property.
- [`void add \(const property_el &el\)`](#)
Add an OPC property.
- [`void add \(const opc_list &o\)`](#)
Add an OPC list.
- [`bool is_published \(\) const`](#)
Is this item published?

- bool [is_READONLY](#) () const
Is readonly?
- bool [get_property](#) (int prop, std::stringcase &val) const
Get string property.
- bool [get_property](#) (int prop, int &val) const
Get integer property.
- bool [get_property](#) (int prop, double &val) const
Get real property.

Protected Attributes

- [opc_enum opc](#)
OPC state.
- [property_map opc_prop](#)
List of OPC properties.

8.37.1 Detailed Description

OPC list.

This class stores OPC properties.

Definition at line 144 of file ParseUtil.h.

The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.38 ParseUtil::optarg Class Reference

Optional arguments.

```
#include <ParseUtil.h>
```

Public Member Functions

- [optarg](#) ()
Default constructor.
- [optarg](#) (const std::stringcase &arg)
- [~optarg](#) ()
Destructor.
- int [parse](#) (const std::stringcase &arg)
- int [argc](#) () const
Returns the number of arguments.
- const char *const * [argv](#) () const
Returns the argument list.
- const bool * [argp](#) () const
Return the processed argument list.
- bool * [argp](#) ()
Return the processed argument list.
- bool [all_DONE](#) () const
Returns true if there all arguments are unprocessed.

Protected Member Functions

- `optarg` (const `optarg` &)
Disabled copy constructor.
- `optarg` & `operator=` (const `optarg` &)
Disabled assignment operator.
- `void setup` (int size)
Setup.

Protected Attributes

- `int mysize`
Size of allocated arrays.
- `int myargc`
Number of arguments.
- `char ** myargv`
Argument list.
- `bool * myargp`
Processed argument list.

8.38.1 Detailed Description

Optional arguments.

This class transforms a string into a standard program argument.

Definition at line 73 of file ParseUtil.h.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 optarg()

```
ParseUtil::optarg::optarg (
    const std::string& arg) [inline], [explicit]
```

Constructor

Parameters

<code>arg</code>	Option argument string
------------------	------------------------

Definition at line 80 of file ParseUtil.h.

References `parse()`.

Here is the call graph for this function:



8.38.3 Member Function Documentation

8.38.3.1 parse()

```
int ParseUtil::optarg::parse (const std::string& arg)
```

Parse string argument

Parameters

<i>arg</i>	Option argument string
------------	------------------------

Returns

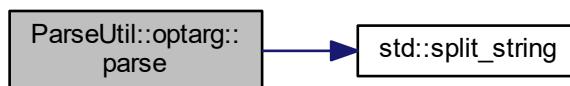
Number of processed arguments

Definition at line 60 of file ParseUtil.cpp.

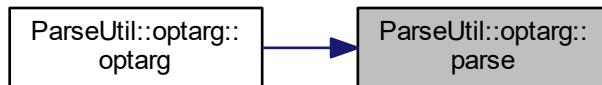
References std::split_string().

Referenced by optarg().

Here is the call graph for this function:



Here is the caller graph for this function:



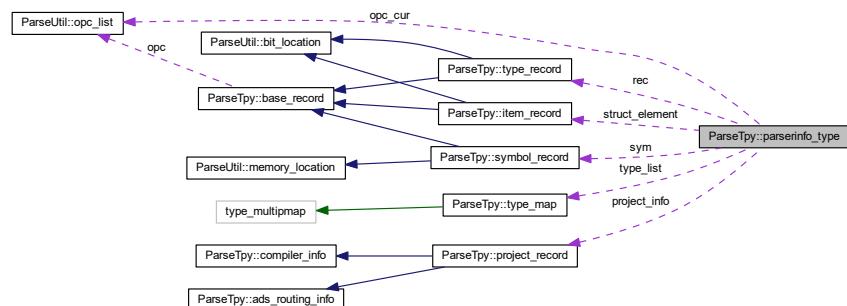
The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.39 ParseTpy::parserinfo_type Class Reference

Parser information.

Collaboration diagram for ParseTpy::parserinfo_type:



Public Member Functions

- [`parserinfo_type \(project_record &p, symbol_list &s, type_map &t\)`](#)
Constructor.
- [`symbol_list & get_symbols \(\)`](#)
Get symbol list.
- [`type_map & get_types \(\)`](#)
Get type list.
- [`project_record & get_projectinfo \(\)`](#)
Get project information.
- [`void init \(\)`](#)
Initialize temporary parser info.
- [`type_enum get_type_description \(\) const`](#)
Get type of parsed object.
- [`bool verytop \(\)`](#)
the very top of the xml tag hierarchy (not within any tag)
- [`bool top \(\)`](#)
the top of the xml tag hierarchy (within the PlcProjectInfo tag)

Public Attributes

- int `ignore`
ignore elements during parsing (with level)
- bool `projects`
parsing withing PlcProjectInfo tag
- int `routing`
parsing within Routing tag (1) or AdsInfo (2), Net ID (3), Port (4), Target name (5)
- int `compiler`
parsing within compiler tag (1) or compiler version (2), Twincat version (3), CPU family (4)
- int `types`
parsing within DataTypes tag (1) or DataType tag (2)
- int `symbols`
parsing within Symbols tag (1) or Symbol tag (2)
- `symbol_record sym`
temporary symbol information during parsing
- `type_record rec`
temporary type information during parsing
- int `name_parse`
- int `type_parse`
level indicator for type parsing
- `opc_list * opc_cur`
Pointer to current opc list (types only)
- `property_el opc_prop`
temporary opc element
- int `opc_parse`
level indicator for opc element
- `std::stringcase opc_data`
temporary data string for parsed opc data
- int `opc_cdata`
temporary cdata indicator
- int `igroup_parse`
level indicator for IGroup
- int `ioffset_parse`
level indicator for IOffset
- int `bitsize_parse`
level indicator for BitSize
- int `bitoffs_parse`
level indicator for BitOffs
- `std::stringcase data`
temporary data string for parsed igrup/ioffset/bitsize/bitoffs
- int `array_parse`
level indicator for array info parsing
- `std::stringcase array_data`
temporary data string for parsed array info
- `dimension array_bounds`
temporary array dimension element
- int `enum_parse`
level indicator for enum parsing
- `std::stringcase enum_data`
temporary data string for parsed enum data

- `enum_pair enum_element`
`temporary enum element`
- `std::stringcase enum_comment`
`temporary enum comment`
- `int struct_parse`
`level indicator for struct parsing`
- `item_record struct_element`
`temporary structure element`
- `int fb_parse`
`level indicator for function block parsing`

Protected Attributes

- `symbol_list * sym_list`
`pointer to symbol list`
- `type_map * type_list`
`pointer to type list`
- `project_record * project_info`
`pointer to project info`

8.39.1 Detailed Description

Parser information.

This structure keeps track of the parser information.

Definition at line 41 of file ParseTpy.cpp.

8.39.2 Member Data Documentation

8.39.2.1 name_parse

```
int ParseTpy::parserinfo_type::name_parse
```

level indicator for type name parsing level indicators in general: 0 - not encountered, 1 - parsed, 2 - currently processing, 3 - further sub tag parsing

Definition at line 97 of file ParseTpy.cpp.

The documentation for this class was generated from the following file:

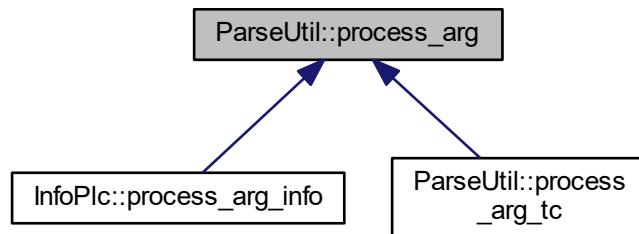
- [ParseTpy.cpp](#)

8.40 ParseUtil::process_arg Class Reference

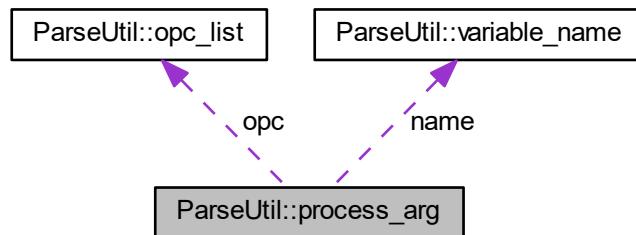
Arguments for processing.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::process_arg:



Collaboration diagram for ParseUtil::process_arg:



Public Member Functions

- `process_arg (const variable_name &vname, process_type_enum pt, const opc_list &o, const std::stringcase &tname, bool at)`
- `const variable_name & get_var () const`
Get variable.
- `const std::stringcase & get_name () const`
Get name.
- `const std::stringcase & get_alias () const`
Get alias.
- `const std::stringcase & get_type_name () const`
Get type name.

- const `opc_list & get_opc () const`
Get OPC list.
- `process_type_enum get_process_type () const`
Get process type.
- `std::stringcase get_process_string () const`
Get process type string.
- bool `is_atomic () const`
Is atomic (or structured) type.
- virtual `std::stringcase get_full () const =0`

Protected Attributes

- const `variable_name & name`
name of type
- const `std::stringcase & type_n`
type definition
- const `opc_list & opc`
list of opc properties
- `process_type_enum ptype`
Process Type.
- bool `atomic`
Atomic element.

8.40.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 343 of file ParseUtil.h.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 process_arg()

```
ParseUtil::process_arg::process_arg (
    const variable_name & vname,
    process_type_enum pt,
    const opc_list & o,
    const std::stringcase & tname,
    bool at ) [inline]
```

Constructor

Parameters

<i>vname</i>	Variable name
<i>pt</i>	Process type
<i>o</i>	OPC list
<i>tname</i>	Type name
<i>at</i>	Atomic type

Definition at line 352 of file ParseUtil.h.

8.40.3 Member Function Documentation

8.40.3.1 get_full()

```
virtual std::stringcase ParseUtil::process_arg::get_full ( ) const [pure virtual]
```

Gets a string representation of a PLC & memory location

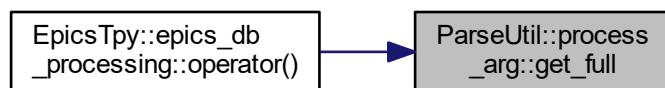
Returns

string with format "prefixigroup/ioffset:size", empty on error

Implemented in [ParseUtil::process_arg_tc](#), and [InfoPlc::process_arg_info](#).

Referenced by [EpicsTpy::epics_db_processing::operator\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

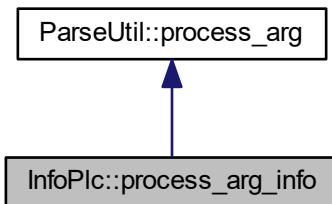
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.41 InfoPlc::process_arg_info Class Reference

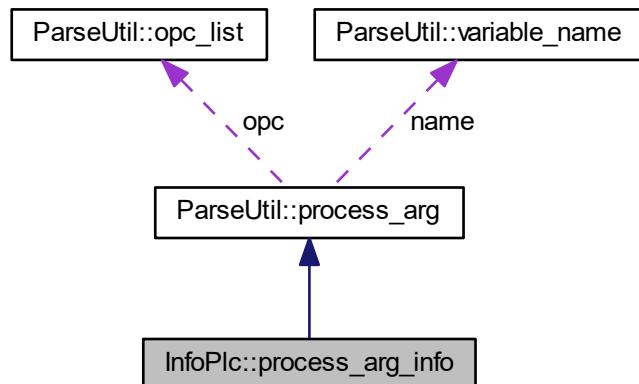
Arguments for processing.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::process_arg_info:



Collaboration diagram for InfoPlc::process_arg_info:



Public Member Functions

- `process_arg_info (const ParseUtil::variable_name &vname, ParseUtil::process_type_enum pt, const ParseUtil::opc_list &o, const std::stringcase &tname, bool at, const std::stringcase &plcaddr)`
- `virtual std::stringcase get_full () const`

Protected Attributes

- `const std::stringcase & tcplc_addr`
TwinCAT PLC address.

8.41.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 218 of file infoPlc.h.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 process_arg_info()

```
InfoPlc::process_arg_info::process_arg_info (
    const ParseUtil::variable_name & vname,
    ParseUtil::process_type_enum pt,
    const ParseUtil::opc_list & o,
    const std::stringcase & tname,
    bool at,
    const std::stringcase & plcaddr ) [inline]
```

Constructor

Parameters

<i>vname</i>	Variable name
<i>pt</i>	Process type
<i>o</i>	OPC list
<i>tname</i>	Type name
<i>at</i>	Atomic type
<i>plcaddr</i>	TwinCAT PLC address

Definition at line 228 of file infoPlc.h.

8.41.3 Member Function Documentation

8.41.3.1 get_full()

```
std::stringcase InfoPlc::process_arg_info::get_full () const [virtual]
```

Gets a string representation of a PLC & memory location

Returns

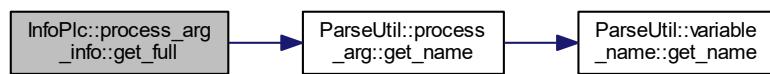
string with format "prefixigroup/offset:size", empty on error

Implements [ParseUtil::process_arg](#).

Definition at line 1222 of file [infoPlc.cpp](#).

References [ParseUtil::process_arg::get_name\(\)](#), and [tcplc_addr](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

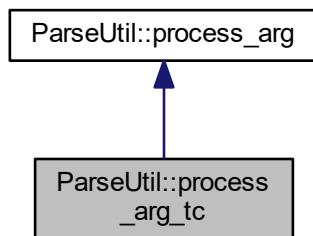
- [infoPlc.h](#)
- [infoPlc.cpp](#)

8.42 ParseUtil::process_arg_tc Class Reference

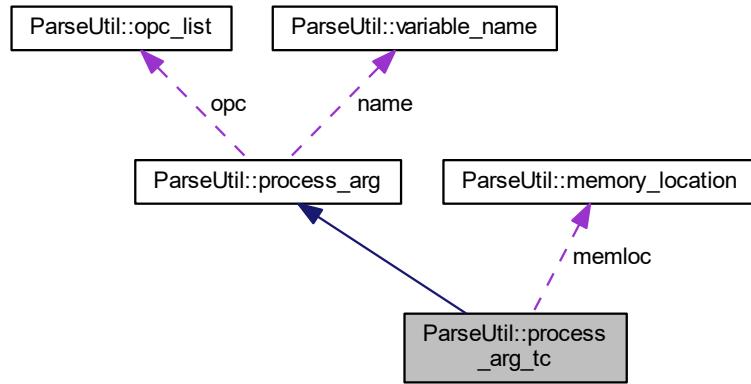
Arguments for processing.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::process_arg_tc:



Collaboration diagram for ParseUtil::process_arg_tc:



Public Member Functions

- `process_arg_tc (const memory_location &loc, const variable_name &vname, process_type_enum pt, const opc_list &o, const std::stringcase &tname, bool at)`
- `int get_igroup () const`
Get IGroup.
- `int get_ioffset () const`
Get IOffset.
- `int get_bytesize () const`
Get BitSize.
- `std::stringcase get () const`
- `virtual std::stringcase get_full () const`

Protected Attributes

- `const memory_location & memloc`
memory location

8.42.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 395 of file ParseUtil.h.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 process_arg_tc()

```
ParseUtil::process_arg_tc::process_arg_tc (
    const memory_location & loc,
    const variable_name & vname,
    process_type_enum pt,
    const opc_list & o,
    const std::stringcase & tname,
    bool at ) [inline]
```

Constructor

Parameters

<i>loc</i>	Memory location
<i>vname</i>	Variable name
<i>pt</i>	Process type
<i>o</i>	OPC list
<i>tname</i>	Type name
<i>at</i>	Atomic type

Definition at line 405 of file ParseUtil.h.

8.42.3 Member Function Documentation

8.42.3.1 get()

```
std::stringcase ParseUtil::process_arg_tc::get () const [inline]
```

Gets a string representation of a memory location

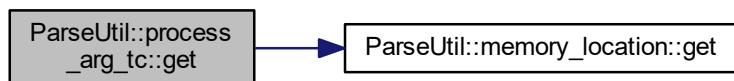
Returns

string with format "igroup/offset:size", empty on error

Definition at line 418 of file ParseUtil.h.

References ParseUtil::memory_location::get(), and memloc.

Here is the call graph for this function:



8.42.3.2 get_full()

```
std::string ParseUtil::process_arg_tc::get_full () const [virtual]
```

Gets a string representation of a PLC & memory location

Returns

string with format "prefix:group/offset:size", empty on error

Implements [ParseUtil::process_arg](#).

Definition at line 323 of file ParseUtil.cpp.

References ParseUtil::OPC_PROP_PLCNAME.

The documentation for this class was generated from the following files:

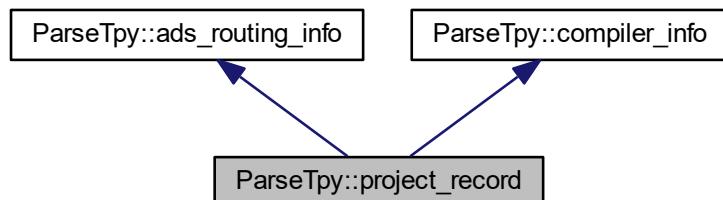
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.43 ParseTpy::project_record Class Reference

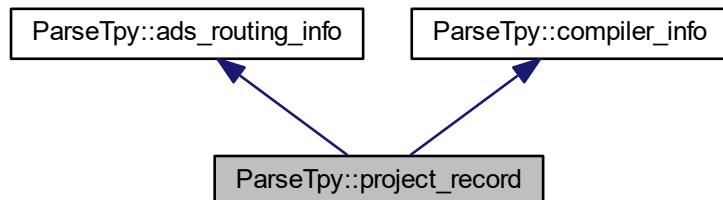
Project information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::project_record:



Collaboration diagram for ParseTpy::project_record:



Public Member Functions

- [project_record \(\)](#)

Default constructor.

Additional Inherited Members

8.43.1 Detailed Description

Project information.

This is a base class for storing the project information

Definition at line 137 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.44 DevTc::register_devsup Class Reference

Device support registration.

```
#include <devTc.h>
```

Collaboration diagram for DevTc::register_devsup:

DevTc::register_devsup  the_register_devsup

Public Types

- [typedef auto link_func\(dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord\) -> bool](#)
Type describing the link function.
- [typedef std::pair< std::regex, link_func & > test_pattern](#)
pair of pattern and link function
- [typedef std::vector< test_pattern > test_pattern_list](#)
list of pattern/link functions

Static Public Member Functions

- static void `add` (const std::regex &rgx, `link_func` &func)
Register a pattern/link function.
- static bool `linkRecord` (const std::stringcase &inpout, dbCommon *pEpicsRecord, `plc::BaseRecordPtr` &p← Record)
linkRecord

Protected Member Functions

- `register_devsup ()`
Default constructor (adds linkTcRecord entry)
- `register_devsup (const register_devsup &)`
Disabled copy constructor.
- `register_devsup & operator= (const register_devsup &)`
Disabled assignment operator.

Protected Attributes

- `test_pattern_list tp_list`
list of pattern and links

Static Protected Attributes

- static `register_devsup the_register_devsup`
the one global instance of the register class

8.44.1 Detailed Description

Device support registration.

This is a class for managing device support for multiple record types, such as TwinCAT/ADS and Info.

Definition at line 41 of file devTc.h.

The documentation for this class was generated from the following files:

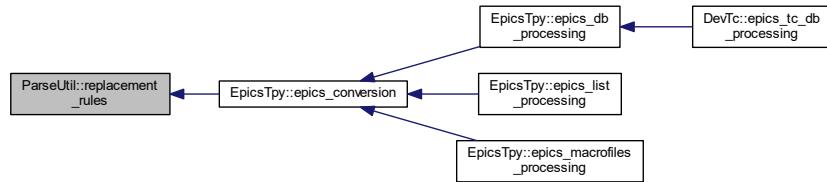
- `devTc.h`
- `devTc.cpp`

8.45 ParseUtil::replacement_rules Class Reference

Replacement rules.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::replacement_rules:



Public Member Functions

- `replacement_rules (bool rec=true)`
Default constructor.
- `replacement_rules (const replacement_table &t, bool rec=true)`
Constructor.
- `void add_rule (const std::stringcase &var, const std::stringcase &val)`
Add a rule.
- `void set_rule_table (const replacement_table &t)`
set table
- `replacement_table & get_rule_table ()`
get table
- `const replacement_table & get_rule_table () const`
get table
- `std::stringcase apply_replacement_rules (const std::stringcase &s) const`
replace
- `bool HasRules () const`
Has rules.
- `bool is_recursive () const`
Is recursive?
- `void set_recursive (bool rec)`
Set recursive.

Static Public Attributes

- `static const char *const prefix = "${"`
prefix for replacement rule: \${
- `static const char *const suffix = "}"`
suffix for replacement rule: }

Protected Attributes

- `replacement_table table`
Replacement table.
- `bool recursive`
Recursive replacement.

8.45.1 Detailed Description

Replacement rules.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 27 of file ParseUtil.h.

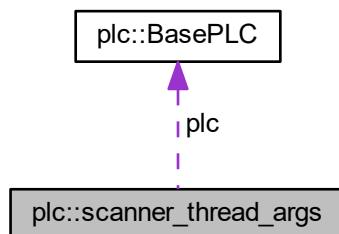
The documentation for this class was generated from the following files:

- `ParseUtil.h`
- `ParseUtil.cpp`

8.46 plc::scanner_thread_args Struct Reference

Scanner thread arguments.

Collaboration diagram for plc::scanner_thread_args:



Public Attributes

- `plc::BasePLC * plc`
PLC this scanner operates on.
- `long scanperiod`
Period in ms of the scanner.
- `plc::BasePLC::scanner_func scanner`
Address of the scanner function to be used (read, write, or update)

8.46.1 Detailed Description

Scanner thread arguments.

Structure for arguments sent to a scanner thread

Definition at line 660 of file plcBase.cpp.

The documentation for this struct was generated from the following file:

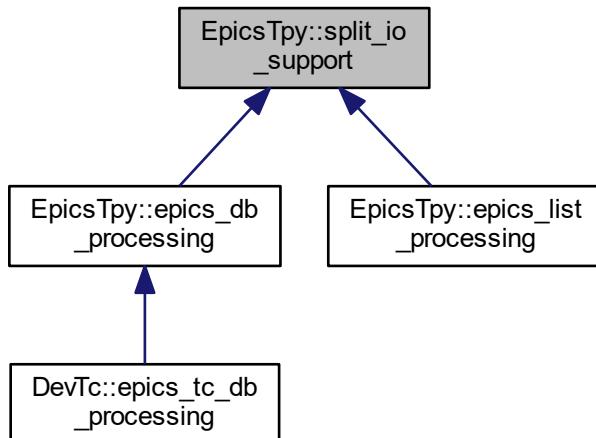
- [plcBase.cpp](#)

8.47 EpicsTpy::split_io_support Class Reference

Split IO support.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::split_io_support:



Public Member Functions

- [split_io_support \(\)](#)
Default constructor.
- [split_io_support \(const std::string& fname, bool split=false, int max=0\)](#)
Constructor.
- [split_io_support \(const std::string& fname, int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)
- [~split_io_support \(\)](#)
Destructor.
- [split_io_support \(const split_io_support &\)](#)

- `split_io_support & operator= (const split_io_support &)`
- `bool operator! () const`

Return error.
- `bool increment (bool readonly)`
- `void flush ()`

Flush contents of output files.
- `FILE * get_file () const`

Get output file.
- `const std::stringcase & get_filename () const`

Get output filename.
- `bool is_split () const`

Is output split?
- `int get_max () const`

Maximum of channels per file.
- `int get_processed_total () const`

Get number of processed channels.
- `int get_processed_READONLY () const`

Get number of processed readonly channels.
- `int get_processed_IO () const`

Get number of processed input/output channels.

Protected Member Functions

- `void set_filename (const std::stringcase &fname)`

Set output filename.
- `void close ()`

Close files.
- `void set_split (bool split)`

set split
- `void set_max (int max)`

Set maximum of channels per file.
- `int getopt (int argc, const char *const argv[], bool argp[] = 0)`

Protected Attributes

- `bool error`

Error.
- `std::stringcase outfilename`

Output filename.
- `bool split_io`

Split output into read only channels and input/output channels.
- `int split_n`

Maximum number of channels per file; 0 indicates no limit.
- `FILE * outf`

Output file.
- `FILE * outf_in`

Output file for read only channels.
- `FILE * outf_io`

Output file for input/output channels.

- int `rec_num`
Current number of processed channels (records)
- int `rec_num_in`
Current number of processed read only channels (records)
- int `rec_num_io`
Current number of processed input/output channels (records)
- int `file_num_in`
Current file number of processed read only channels (records)
- int `file_num_io`
Current file number of processed input/output channels (records)
- `std::stringcase file_num_in_s`
Contains the readonly file number in string format.
- `std::stringcase file_num_io_s`
Contains the input/output file number in string format.
- `std::stringcase file_in_s`
Contains the file extenstion for readonly files.
- `std::stringcase file_io_s`
Contains the file extenstion for input/output files.

8.47.1 Detailed Description

Split IO support.

Split file IO support Output can be split in multiple files if the number of channels exceeds the maximum specified for a file

Definition at line 162 of file TpyToEpics.h.

8.47.2 Constructor & Destructor Documentation

8.47.2.1 `split_io_support()` [1/2]

```
EpicsTpy::split_io_support::split_io_support (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[ ] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

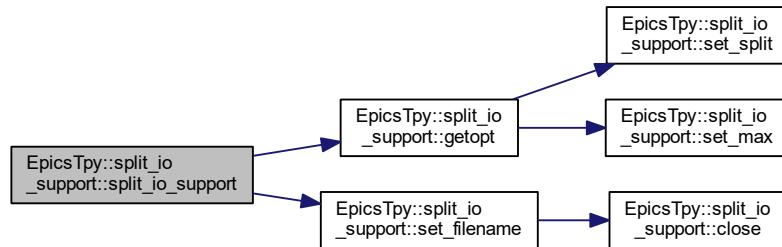
Parameters

<code>fname</code>	Filename for output
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 187 of file TpyToEpics.h.

References getopt(), and set_filename().

Here is the call graph for this function:



8.47.2.2 split_io_support() [2/2]

```
EpicsTpy::split_io_support::split_io_support (
    const split_io_support & iosup )
```

Copy constructor File pointers will be moved over and the original ones will become invalid.

Definition at line 200 of file TpyToEpics.cpp.

8.47.3 Member Function Documentation

8.47.3.1 getopt()

```
int EpicsTpy::split_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [protected]
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/ysio: Splits database into input only and input/output records /nsio: Does not split database by record type (default) /sn 'num': Splits database or listing into files with no more than num records /sn 0: Does not split database or listing into multiple files (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

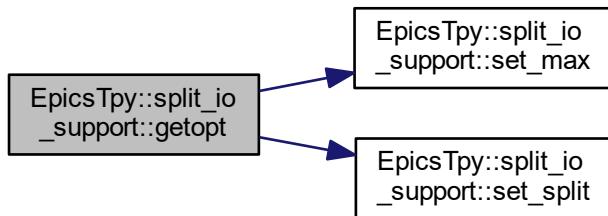
Number of arguments processed

Definition at line 238 of file TpyToEpics.cpp.

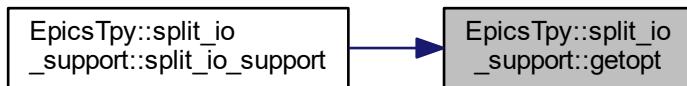
References [set_max\(\)](#), and [set_split\(\)](#).

Referenced by [split_io_support\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.47.3.2 increment()**

```
bool EpicsTpy::split_io_support::increment (
    bool readonly )
```

Increase the channel number

Parameters

<i>readonly</i>	Indicates if channel is readonly
-----------------	----------------------------------

Returns

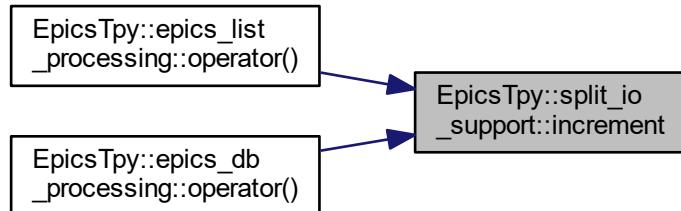
True if no error

Definition at line 368 of file TpyToEpics.cpp.

References error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

Here is the caller graph for this function:

**8.47.3.3 operator=()**

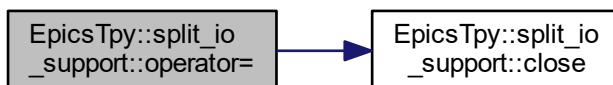
```
split_io_support & EpicsTpy::split_io_support::operator= (
    const split_io_support & iosup )
```

Assignment File pointers will be moved over and the original ones will become invalid.

Definition at line 211 of file TpyToEpics.cpp.

References close(), error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

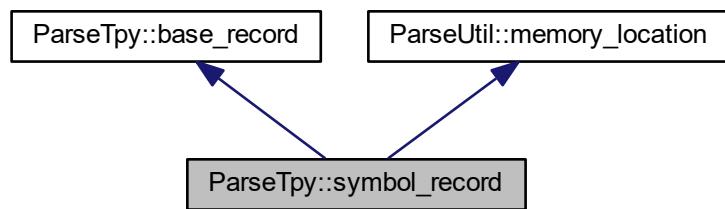
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.48 ParseTpy::symbol_record Class Reference

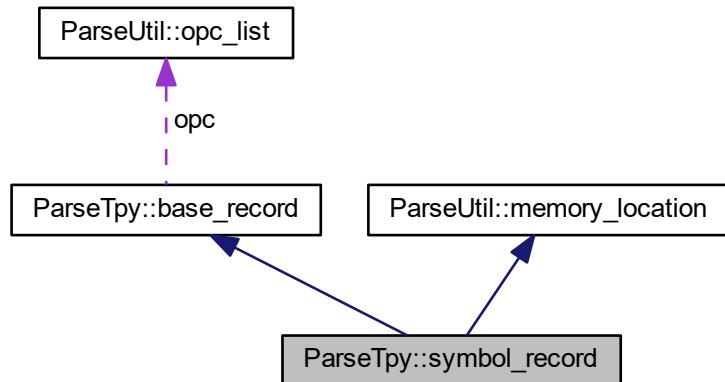
Symbol record.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::symbol_record:



Collaboration diagram for ParseTpy::symbol_record:



Public Member Functions

- [symbol_record \(\)](#)
Default constructor.
- [const memory_location & get_location \(\) const](#)
get memory location
- [memory_location & get_location \(\)](#)
get memory location

Additional Inherited Members

8.48.1 Detailed Description

Symbol record.

This structure holds a symbol record

Definition at line 340 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.49 syminfo_processing Class Reference

Symbol processing.

Public Member Functions

- [`syminfo_processing`](#) (`FILE *outfile=0, bool atomic=true`)
Constructor.
- [`bool operator\(\)`](#) (`const process_arg &arg`)
Process.

Protected Attributes

- `FILE * outf`
Output file.
- `bool firstline`
Firstline?

8.49.1 Detailed Description

Symbol processing.

Symbol processing

Definition at line 17 of file ParseTpyInfo.cpp.

The documentation for this class was generated from the following file:

- [ParseTpyInfo.cpp](#)

8.50 plc::System Class Reference

[System](#) to keep track of PLCs.

```
#include <plcBase.h>
```

Public Types

- `typedef std::recursive_mutex mutex_type`
Defines the mutex type.
- `typedef std::lock_guard< mutex_type > guard`
Defined the mutex guard type.

Public Member Functions

- `bool add (BasePLC *plc)`
- `bool add (BasePLCPtr plc)`
- `BasePLCPtr find (std::stringcase id)`
Finds a PLC by its name.
- template<typename func >
`void for_each (func &f)`
- `void printVals ()`
Print all PLC record values to the console.
- `void printVal (const std::string &var)`
- `void start ()`
Start scanning after ioc is running.
- `void stop ()`
Stop scanning when ioc is paused.
- `bool is_ioc_running () const`
get loc run state
- `void set_ioc_state (bool run)`
set loc run state

Static Public Member Functions

- `static System & get ()`
Return a reference to the gloabl System variable.

Protected Attributes

- `mutex_type mux`
Mutex to synchronize access to this class.
- `BasePLCList PLCs`
Master list of all PLCs.
- `bool locRun`
IOC is running.

8.50.1 Detailed Description

[System](#) to keep track of PLCs.

This is a class for managing multiple PLCs.

Definition at line 899 of file plcBase.h.

8.50.2 Member Function Documentation

8.50.2.1 add() [1/2]

```
bool plc::System::add (
    BasePLC * plc ) [inline]
```

Add a new PLC, PLC will be adopted

Parameters

<i>plc</i>	Pointer to plc
------------	----------------

Returns

true if successful

Definition at line 912 of file plcBase.h.

8.50.2.2 add() [2/2]

```
bool plc::System::add (
    BasePLCPtr plc )
```

Add a new PLC

Parameters

<i>plc</i>	Smart pointer to plc
------------	----------------------

Returns

true if successful

Definition at line 786 of file plcBase.cpp.

References mux, and PLCs.

8.50.2.3 for_each()

```
template<typename func >
void plc::System::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

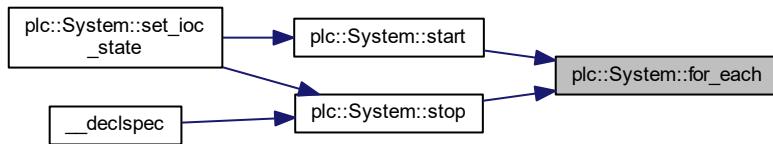
<code>f</code>	Function which takes <code>BaseRecord*</code> as the argument
----------------	---

Definition at line 216 of file `plcBaseTemplate.h`.

References mux, and PLCs.

Referenced by `start()`, and `stop()`.

Here is the caller graph for this function:



8.50.2.4 printVal()

```
void plc::System::printVal (
    const std::string & var )
```

Print PLC record value to the console

Parameters

<code>var</code>	Variable name or wildcard
------------------	---------------------------

Definition at line 811 of file `plcBase.cpp`.

References mux, and PLCs.

Referenced by `DevTc::tcPrintVal()`.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

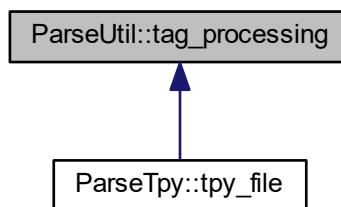
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.51 ParseUtil::tag_processing Class Reference

Tag processing selection.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::tag_processing:



Public Member Functions

- [`tag_processing \(\)`](#)
Default constructor.
- [`tag_processing \(bool all, process_tag_enum proctags, bool nostring=false\)`](#)
- [`tag_processing \(int argc, const char *const argv\[\], bool argp\[\] = 0\)`](#)
- [`int getopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\)`](#)
- [`bool get_export_all \(\) const`](#)
Get the export rule.
- [`void set_export_all \(bool all\)`](#)

- `process_tag_enum get_process_tags () const`
Get the process rule.
- `void set_process_tags (process_tag_enum proctags)`
Set the process rule.
- `bool get_no_strings () const`
Get the string rule.
- `void set_no_strings (bool nostring)`
Set the string rule.

Protected Attributes

- `bool export_all`
Process all symbols regardless of opc publish setting.
- `process_tag_enum process_tags`
Process only atomic types.
- `bool no_string_tags`
Don't process strings.

8.51.1 Detailed Description

Tag processing selection.

Class to specify which symbols and tags/names to process

Definition at line 444 of file ParseUtil.h.

8.51.2 Constructor & Destructor Documentation

8.51.2.1 tag_processing() [1/2]

```
ParseUtil::tag_processing::tag_processing (
    bool all,
    process_tag_enum proctags,
    bool nostring = false ) [inline]
```

Constructor

Parameters

<code>all</code>	Process all tags
<code>proctags</code>	Process atomic and/or structured tags
<code>nostring</code>	Don't process string tags

Definition at line 454 of file ParseUtil.h.

8.51.2.2 tag_processing() [2/2]

```
ParseUtil::tag_processing::tag_processing (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Commaline arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

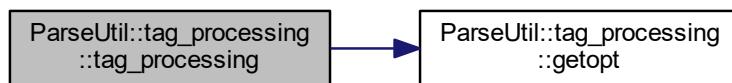
Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 465 of file ParseUtil.h.

References getopt().

Here is the call graph for this function:



8.51.3 Member Function Documentation

8.51.3.1 getopt()

```
int ParseUtil::tag_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/ea: Export all variables regardless of OPC setting /eo: Only export variables which are marked as OPC export (default) /ns: No string variables are processed /ys: String variables are processes (default) /pa: Call process for all types (default) /ps: Call process for simple (atomic) types only /pc: Call process for complex (structure and array) types only

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

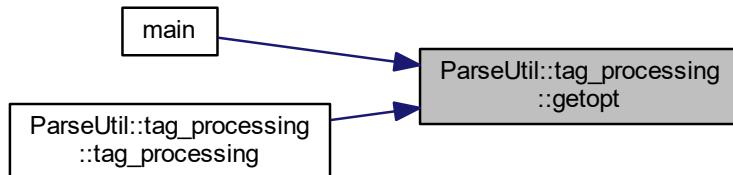
Number of arguments processed

Definition at line 339 of file ParseUtil.cpp.

References ParseUtil::process_all, ParseUtil::process_atomic, and ParseUtil::process_structured.

Referenced by main(), and tag_processing().

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

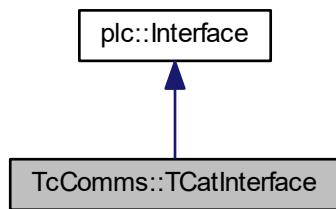
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.52 TComms::TCatInterface Class Reference

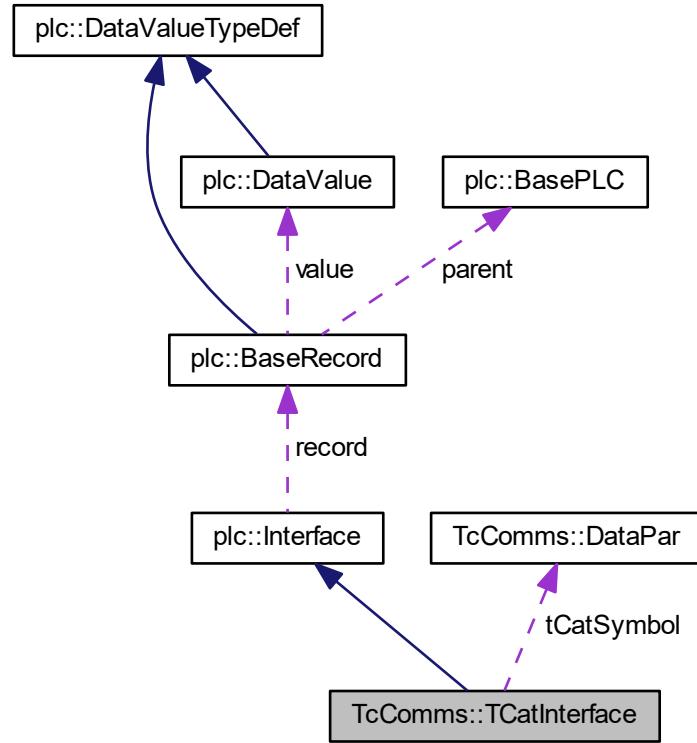
TCat interface class.

```
#include <tcomms.h>
```

Inheritance diagram for TComms::TCatInterface:



Collaboration diagram for TcComms::TCatInterface:



Public Member Functions

- **TCatInterface (plc::BaseRecord &dval)**
Constructor.
- **TCatInterface (plc::BaseRecord &dval, const std::stringcase &name, unsigned long group, unsigned long offset, unsigned long length, const std::stringcase &type, bool isStruct, bool isEnum)**
- **~TCatInterface ()**
Deconstructor.
- **const std::stringcase get_tCatName () const**
Get name of TCat symbol.
- **void set_tCatName (std::stringcase name)**
Set name of TCat symbol.
- **virtual const char * get_symbol_name () const**
Get symbol name.
- **const std::stringcase get_tCatType ()**
Get TCat data type.
- **void set_tCatType (std::stringcase type)**
Set TCat data type.
- **DataPar get_tCatSymbol ()**
Get structure containing index group, index offset, size.
- **unsigned long get_indexGroup () const**

- `void set_indexGroup (unsigned long group)`

Get index group.
- `unsigned long get_indexOffset () const`

Set index group.
- `void set_indexOffset (unsigned long offset)`

Get index offset.
- `unsigned long get_size () const`

Get size in bytes of symbol.
- `void set_size (unsigned long nBytes)`

Set size in bytes of symbol.
- `size_t get_requestOffs () const`

Get offset into response buffer.
- `void set_requestOffs (size_t pVal)`

Set offset into response buffer.
- `TcPLC * get_parent ()`

Get parent PLC that owns this record.
- `const TcPLC * get_parent () const`

Get parent PLC that owns this record.
- `int get_requestNum ()`

Get the request group number this record is in.
- `void set_requestNum (int rNum)`

Set the request group number this record is in.
- `virtual void printVal (FILE *fp)`
- `virtual bool push () override`

Does nothing.
- `virtual bool pull () override`

Does nothing.

Protected Attributes

- `std::stringcase tCatName`

Name of TCat symbol.
- `std::stringcase tCatType`

Data type in TCat.
- `DataPar tCatSymbol`

Struct storing index group, index offset, and length of TC symbol.
- `int requestNum`

Which request group in the PLC.
- `size_t requestOffs`

Offset into response buffer.

8.52.1 Detailed Description

TCat interface class.

This is a class for a TCat interface

Definition at line 70 of file tcComms.h.

8.52.2 Constructor & Destructor Documentation

8.52.2.1 TCatInterface()

```
TcComms::TCatInterface::TCatInterface (
    plc::BaseRecord & dval,
    const std::string& name,
    unsigned long group,
    unsigned long offset,
    unsigned long length,
    const std::string& type,
    bool isStruct,
    bool isEnum )
```

Constructor

Parameters

<i>dval</i>	BaseRecord that this interface is part of
<i>name</i>	Name of TCat symbol
<i>group</i>	Index group of TCat symbol
<i>offset</i>	Index offset of TCat symbol
<i>length</i>	Size in bytes of TCat symbol
<i>type</i>	Name of TCat data type
<i>isStruct</i>	True = this symbol is a structure in TCat
<i>isEnum</i>	True = this symbol is an enum in TCat

Definition at line 88 of file tcComms.cpp.

8.52.3 Member Function Documentation

8.52.3.1 printVal()

```
void TcComms::TCatInterface::printVal (
    FILE * fp ) [virtual]
```

Prints TCat symbol value and information

Parameters

<i>fp</i>	File to print symbol to
-----------	-------------------------

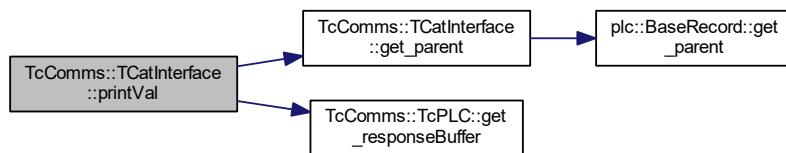
This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file fp.

Reimplemented from [plc::Interface](#).

Definition at line 132 of file tcComms.cpp.

References `get_parent()`, `TcComms::TcPLC::get_responseBuffer()`, `TcComms::DataPar::length`, `requestNum`, `requestOffs`, `tCatName`, `tCatSymbol`, and `tCatType`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

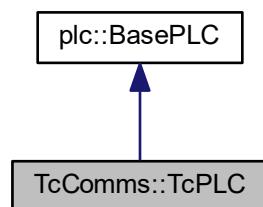
- [tcComms.h](#)
- [tcComms.cpp](#)

8.53 TcComms::TcPLC Class Reference

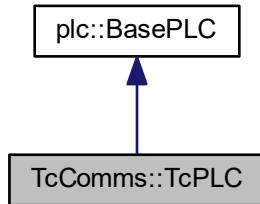
TwinCAT PLC.

```
#include <tccomms.h>
```

Inheritance diagram for TcComms::TcPLC:



Collaboration diagram for TcComms::TcPLC:



Public Types

- `typedef char buffer_type`
Buffer type.
- `typedef std::shared_ptr< buffer_type > buffer_ptr`
Smart pointer to buffer.

Public Member Functions

- `TcPLC (std::string tpyPath)`
Constructor.
- `~TcPLC ()`
Destructor.
- `bool is_valid_tpy ()`
Is typ still valid? Meaning, it hasn't changed.
- `AmsAddr get_addr () const`
Get AMS netID of TwinCAT system and port number for this PLC.
- `bool set_addr (std::stringcase netid, int port)`
- `long get_nReadPort () const`
Get read port number.
- `long get_nWritePort () const`
Get write port number.
- `int get_read_scanner_multiple () const`
Get slowdown multiple for EPICS read.
- `void set_read_scanner_multiple (int mult)`
Set slowdown multiple for EPICS read.
- `ADSSTATE get_ads_state () const`
Get ADS state.
- `bool is_read_active () const`
Is read scanner active and successful.
- `const std::string & get_tpyfilename () const`
Get the tpy filename.
- `bool is_tpyfile_valid () const`
Is the tpy file valid?

- `time_t get_tpyfile_time () const`
Get the file modification time of the tpy file.
- `virtual bool start ()`
Starts the appropriate scanners.
- `bool optimizeRequests ()`
- `buffer_ptr get_responseBuffer (size_t idx)`
- `virtual void printAllRecords ()`
Prints symbol information for entire list of symbols to console.
- `virtual void printRecord (const std::string &var)`

Protected Member Functions

- `virtual void read_scanner ()`
Makes read requests to ADS, makes PlcWrite on all data values.
- `virtual void write_scanner ()`
Collects records to be written to TCat, makes write request.
- `virtual void update_scanner ()`
Makes sure we don't have stale values.
- `void set_ads_state (ADSSTATE state)`
Set ADS state.
- `void setup_ads_notification ()`
Set up ADS status change notification.
- `void remove_ads_notification ()`
Remove ADS status change notification.
- `long openPort ()`
Opens a new ADS communication port.
- `void closePort (long nPort)`

Protected Attributes

- `std::mutex sync`
Mutex.
- `AmsAddr addr`
AMS netID of TwinCAT system and port number for this PLC.
- `std::string pathTpy`
The path of the tpy file.
- `time_t timeTpy`
Modification time of file.
- `std::atomic< bool > checkTpy`
need to check modification time to make sure tpy file hasn't changed
- `bool validTpy`
tpy file is valid and hasn't changed
- `int nRequest`
Number of read request groups.
- `std::vector< DataPar > adsGroupReadRequestVector`
Vector of index group, index offset, size for read requests.
- `std::vector< buffer_ptr > adsResponseBufferVector`
Vector of buffers for each read request group.
- `plc::BaseRecordList nonTcRecords`

- List of all records that don't interface directly with a PLC (info)*
- int `scanRateMultiple`
Slowdown multiple for EPICS read.
 - int `cyclesLeft`
 - int `update_workload`
Workload for update scanner.
 - `plc::BaseRecordPtr update_last`
last updated record
 - `std::atomic<ADSSTATE> ads_state`
ADS state.
 - unsigned long `ads_handle`
ADS handle.
 - `std::atomic<bool> ads_restart`
ADS restart.
 - long `nReadPort`
Port number for ADS read connection.
 - long `nWritePort`
Port number for ADS write connection.
 - long `nNotificationPort`
Port number for ADS notification connection.
 - bool `read_active`
read active and successful

Friends

- void __stdcall `ADScallback` (AmsAddr *, AdsNotificationHeader *, unsigned long)
Notification callback is a friend.

8.53.1 Detailed Description

TwinCAT PLC.

Class for a connection to a TwinCAT PLC This class is derived from a BasePLC object, and specializes in managing records that contain a plc interface for TCat. This class is initialized using a .tpy file, from which it will obtain the AMS address information for connecting with TCat through ADS.

Reading and writing from/to ADS will be managed by this class, with read requests being grouped by continuous memory region in TCat to optimize read scanning for speed. Write requests are made using an ADS sum request.

There is also an option to send a request to ADS to check the status of both the PLC device and also the ADS connection.

Definition at line 257 of file tcComms.h.

8.53.2 Member Function Documentation

8.53.2.1 closePort()

```
void TcComms::TcPLC::closePort (
    long nPort ) [protected]
```

Closes an ADS communication port

Parameters

<code>nPort</code>	Number of port to close
--------------------	-------------------------

Definition at line 858 of file tcComms.cpp.

Referenced by `remove_ads_notification()`.

Here is the caller graph for this function:

**8.53.2.2 `get_responseBuffer()`**

```
TcPLC::buffer_ptr TcComms::TcPLC::get_responseBuffer (
    size_t idx )
```

Get pointer to the beginning of a read request response buffer

Parameters

<code>idx</code>	Index of response buffer
------------------	--------------------------

Returns

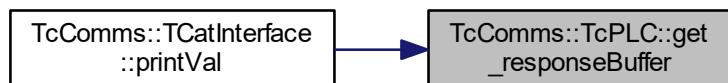
pointer to buffer

Definition at line 544 of file tcComms.cpp.

References `adsResponseBufferVector`.

Referenced by `TcComms::TCatInterface::printVal()`.

Here is the caller graph for this function:



8.53.2.3 optimizeRequests()

```
bool TcComms::TcPLC::optimizeRequests ( )
```

Sorts read channels into request groups. Will make a new request group for channels not in continuous memory region in TCat. Will create buffers of appropriate size for each read request, and let each TCat record know where in the read response buffer the data for that symbol is.

Returns

true if successful

Definition at line 427 of file tcComms.cpp.

8.53.2.4 printRecord()

```
void TcComms::TcPLC::printRecord (
    const std::string & var ) [virtual]
```

Print a record values to stdout. (override for action)

Parameters

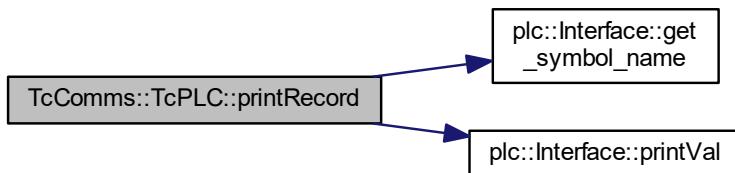
<i>var</i>	variable name (accepts wildcards)
------------	-----------------------------------

Reimplemented from [plc::BasePLC](#).

Definition at line 620 of file tcComms.cpp.

References [plc::Interface::get_symbol_name\(\)](#), [plc::Interface::printVal\(\)](#), and [plc::BasePLC::records](#).

Here is the call graph for this function:



8.53.2.5 set_addr()

```
bool TcComms::TcPLC::set_addr (
    std::stringcase netid,
    int port )
```

Set AMS address

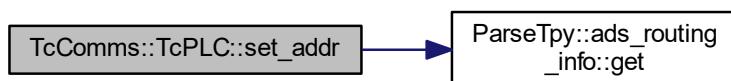
Returns

true if successful

Definition at line 337 of file tcComms.cpp.

References addr, and ParseTpy::ads_routing_info::get().

Here is the call graph for this function:



8.53.2.6 set_ads_state()

```
void TcComms::TcPLC::set_ads_state (
    ADSSTATE state ) [protected]
```

Set ADS state.

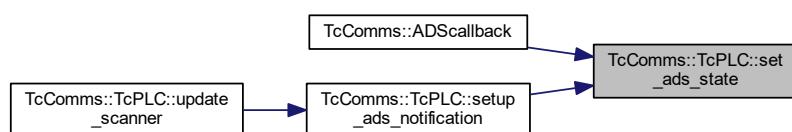
TcPLC::set_ads_state

Definition at line 669 of file tcComms.cpp.

References ads_state, checkTpy, and plc::BasePLC::name.

Referenced by TcComms::ADScallback(), and setup_ads_notification().

Here is the caller graph for this function:



8.53.2.7 update_scanner()

```
void TcComms::TcPLC::update_scanner ( ) [protected], [virtual]
```

Makes sure we don't have stale values.

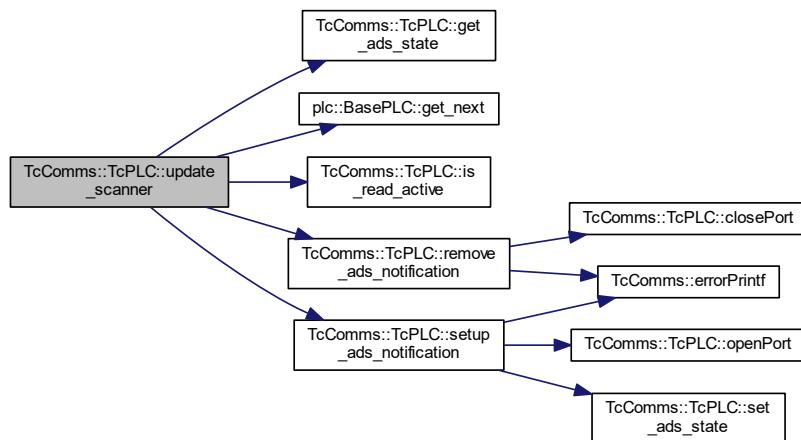
what!?

Reimplemented from [plc::BasePLC](#).

Definition at line 817 of file tcComms.cpp.

References ads_restart, get_ads_state(), plc::BasePLC::get_next(), is_read_active(), plc::BasePLC::mux, plc::BasePLC::name, plc::BasePLC::records, remove_ads_notification(), setup_ads_notification(), update_last, and update_workload.

Here is the call graph for this function:



8.53.3 Friends And Related Function Documentation

8.53.3.1 ADSCallback

```
void __stdcall ADSCallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId ) [friend]
```

Notification callback is a friend.

Callback for ADS state change

Definition at line 648 of file tcComms.cpp.

Referenced by `setup_ads_notification()`.

8.53.4 Member Data Documentation

8.53.4.1 cyclesLeft

`int TcComms::TcPLC::cyclesLeft [protected]`

Cycles until EPICS read will be made Counts down from scanRateMultiple, resets at 0

Definition at line 376 of file tcComms.h.

Referenced by `read_scanner()`.

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

8.54 TcComms::tcProcWrite Class Reference

TwinCAT process write requests.

```
#include <tcComms.h>
```

Public Member Functions

- `tcProcWrite (const AmsAddr &a, long amsport, size_t mrec=1000)`
Default constructor.
- `~tcProcWrite ()`
Destructor: will porcess the TCat writes.
- `tcProcWrite (tcProcWrite &&tp) noexcept`
Move constructor.
- `void operator() (plc::BaseRecord *prec)`
Process on record.
- `void * read_ptr (int sz)`
- `bool add (long igrup, long ioffs, long sz)`

Protected Member Functions

- `bool check_alloc (int extra=0)`
Checks if we have enough memory allocated.
- `void tcwrite ()`
writes the current header/data to TCat
- `tcProcWrite & operator= (tcProcWrite &&) noexcept`
Move operator.

Protected Attributes

- AmsAddr [addr](#)
AMS address.
- long [port](#)
Port to be used to write to TCat.
- char * [ptr](#)
Pointer to header to be written.
- char * [data](#)
Pointer to data to be written.
- size_t [maxrec](#)
Maximum number of individual requests.
- size_t [size](#)
Size of data.
- size_t [alloc](#)
Size of allocated header/data array.
- size_t [count](#)
Current number of individual requests.
- std::vector< [tcProcWrite](#) > [req](#)

8.54.1 Detailed Description

TwinCAT process write requests.

Class for collecting and processing write requests This class iterates through the entire record list on the PLC and collects those records whose data value has a dirty flag set on the plc side. These records are then sent as a group to ADS.

In order to not overload the ADS server, a maximum number of symbols per request is defined, and should not be > 2000.

Definition at line 177 of file tcComms.h.

8.54.2 Member Function Documentation

8.54.2.1 add()

```
bool TcComms::tcProcWrite::add (
    long igrp,
    long ioffs,
    long sz )
```

Add header info

Parameters

<i>igrp</i>	iGroup number for tc write
<i>ioffs</i>	iOffset number for tc write
<i>sz</i>	Size of data to be written

Returns

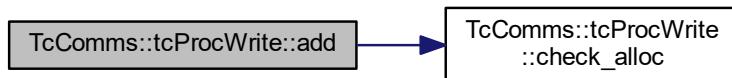
true if successful

Definition at line 244 of file tcComms.cpp.

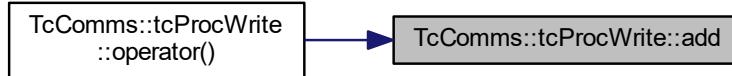
References check_alloc(), count, maxrec, ptr, and req.

Referenced by operator()().

Here is the call graph for this function:



Here is the caller graph for this function:



8.54.2.2 `read_ptr()`

```
void * TcComms::tcProcWrite::read_ptr (
    int sz )
```

Get a pointer to read the value in

Parameters

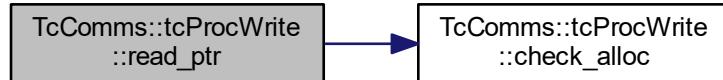
sz	Requested size
----	----------------

Definition at line 232 of file tcComms.cpp.

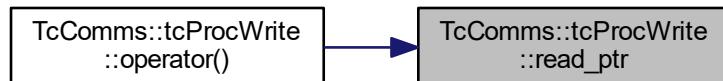
References check_alloc(), data, and size.

Referenced by operator()().

Here is the call graph for this function:



Here is the caller graph for this function:



8.54.3 Member Data Documentation

8.54.3.1 req

```
std::vector<tcProcWrite> TcComms::tcProcWrite::req [protected]
```

Queued TCat requests (each would have reached maxrec individual requests)

Definition at line 224 of file tcComms.h.

Referenced by add(), and ~tcProcWrite().

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

8.55 DevTc::tcRegisterTolocShell Class Reference

Register TC commands.

```
#include <drvTc.h>
```

8.55.1 Detailed Description

Register TC commands.

Register TC commands to IOC shell This class registers the callback functions for the TC IOC commands

Definition at line 18 of file drvTc.h.

The documentation for this class was generated from the following files:

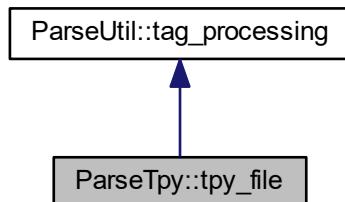
- [drvTc.h](#)
- [drvTc.cpp](#)

8.56 ParseTpy::tpy_file Class Reference

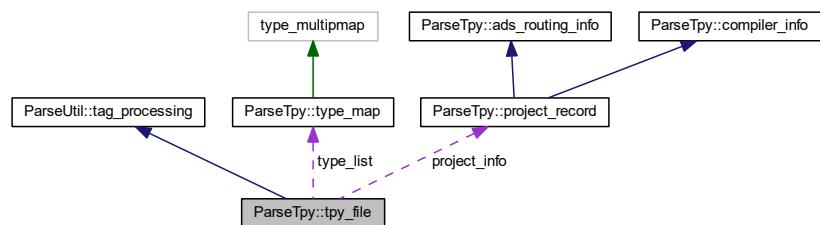
Tpy file parsing.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::tpy_file:



Collaboration diagram for ParseTpy::tpy_file:



Public Member Functions

- `tpy_file ()`
`Default constructor.`
- `tpy_file (FILE *inp)`
`Constructor.`
- `bool parse (FILE *inp)`
`Parse a file.`
- `bool parse (const char *p, int len)`
`Parse a memory region.`
- `const symbol_list & get_symbols () const`
`Return list of symbols.`
- `const type_map & get_types () const`
`Return list of types.`
- `const project_record & get_project_info () const`
`Return project information.`
- template<class Function>
`int process_symbols (Function &process, const std::stringcase &prefix=std::stringcase()) const`
`Process the type tree of a symbol.`
- template<class Function>
`int process_type_tree (const symbol_record &symbol, Function &process, const std::stringcase &prefix=std::stringcase()) const`
`Process the type tree of a symbol.`
- template<class Function>
`int process_type_tree (const type_record &typ, ParseUtil::opc_list defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const`
`Process the type tree of a type.`
- template<class Function>
`int process_type_tree (const std::stringcase &typ, unsigned int id, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const`
`Process the type tree of a type.`

Protected Member Functions

- `void parse_finish ()`
`finish up the parsing`
- template<class Function>
`int process_array (const type_record &typ, dimensions dim, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname, int level) const`
`Process the type tree of a type.`

Protected Attributes

- `project_record project_info`
`Project information.`
- `symbol_list sym_list`
`List of symbols.`
- `type_map type_list`
`List of types.`

8.56.1 Detailed Description

Tpy file parsing.

This class holds the structure of a tpy file

Definition at line 360 of file ParseTpy.h.

8.56.2 Member Function Documentation

8.56.2.1 parse_finish()

```
void ParseTpy::tpy_file::parse_finish ( ) [protected]
```

finish up the parsing

This function is called at the end of parsing. Here we set the TC server name in the OPC variables for each symbol

Definition at line 472 of file ParseTpy.cpp.

References ParseUtil::OPC_PROP_PLCNAME.

8.56.2.2 process_array()

```
template<class Function >
int ParseTpy::tpy_file::process_array (
    const type_record & typ,
    dimensions dim,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname,
    int level ) const [protected]
```

Process the type tree of a type.

Resolves the type information for an array. Calls the process function for each index with an argument of type process_arg.

Parameters

<i>typ</i>	Name of type to resolve
<i>dim</i>	Dimensions of the array
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

Returns

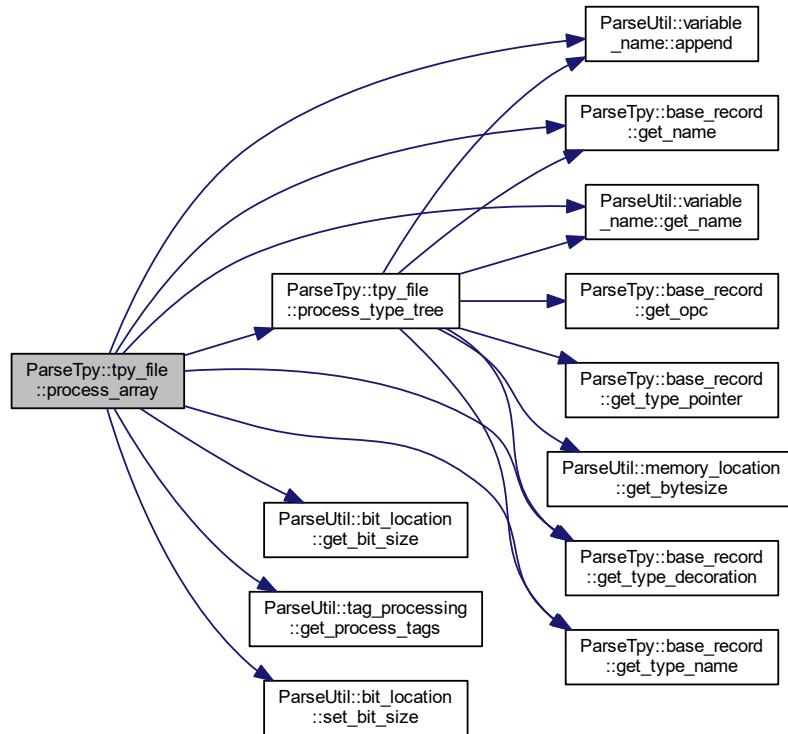
Number of processes variables

Definition at line 218 of file ParseTpyTemplate.h.

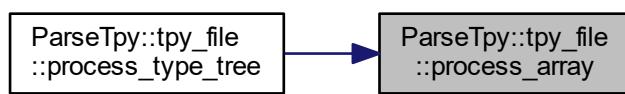
References ParseUtil::variable_name::append(), ParseUtil::bit_location::get_bit_size(), ParseTpy::base_record::get_name(), ParseUtil::variable_name::get_name(), ParseUtil::tag_processing::get_process_tags(), ParseTpy::base_record::get_type_decoration(), ParseTpy::base_record::get_type_name(), ParseUtil::process_all, ParseUtil::process_structured, process_type_tree(), ParseUtil::pt_binary, and ParseUtil::bit_location::set_bit_size().

Referenced by process_type_tree().

Here is the call graph for this function:



Here is the caller graph for this function:



8.56.2.3 process_symbols()

```
template<class Function >
int ParseTpy::tpy_file::process_symbols (
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
```

Process the type tree of a symbol.

Iterates over the symbol list and processes all specified tags.

Parameters

<i>process</i>	Function class
<i>prefix</i>	Prefix which is added to all variable names

Returns

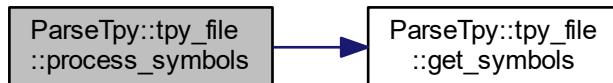
Number of processes variables

Definition at line 9 of file ParseTpyTemplate.h.

References get_symbols().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.56.2.4 process_type_tree() [1/3]

```
template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const symbol_record & symbol,
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
```

Process the type tree of a symbol.

Starts with a symbol and resolves the type information recursively until an atomic type (like INT) is found. Then, calls the process function with an argument of type process_arg. The function must return true if successful and false otherwise.

Parameters

<i>symbol</i>	Symbol to resolve
<i>process</i>	Function class
<i>prefix</i>	Prefix which is added to all variable names

Returns

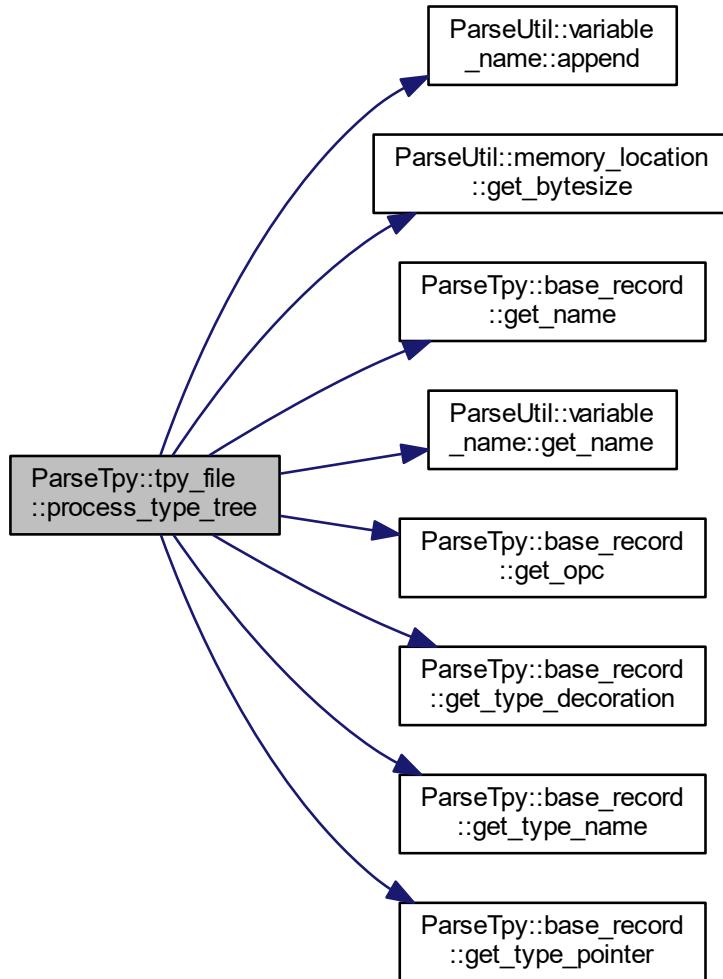
Number of processes variables

Definition at line 27 of file ParseTpyTemplate.h.

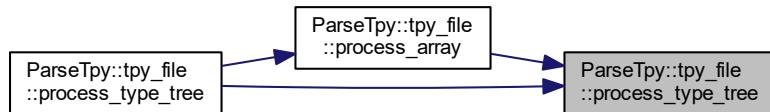
References ParseUtil::variable_name::append(), ParseUtil::memory_location::get_bytesize(), ParseTpy::base_record::get_name(), ParseUtil::variable_name::get_name(), ParseTpy::base_record::get_opc(), ParseTpy::base_record::get_type_decoration(), ParseTpy::base_record::get_type_name(), and ParseTpy::base_record::get_type_pointer().

Referenced by process_array(), and process_type_tree().

Here is the call graph for this function:



Here is the caller graph for this function:



8.56.2.5 process_type_tree() [2/3]

```
template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const type_record & typ,
    ParseUtil::opc_list defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
    int level = 0 ) const
```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like BOOL) is found. Then, calls the process function with an argument of type process_arg.

Parameters

<i>typ</i>	Type to resolve
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

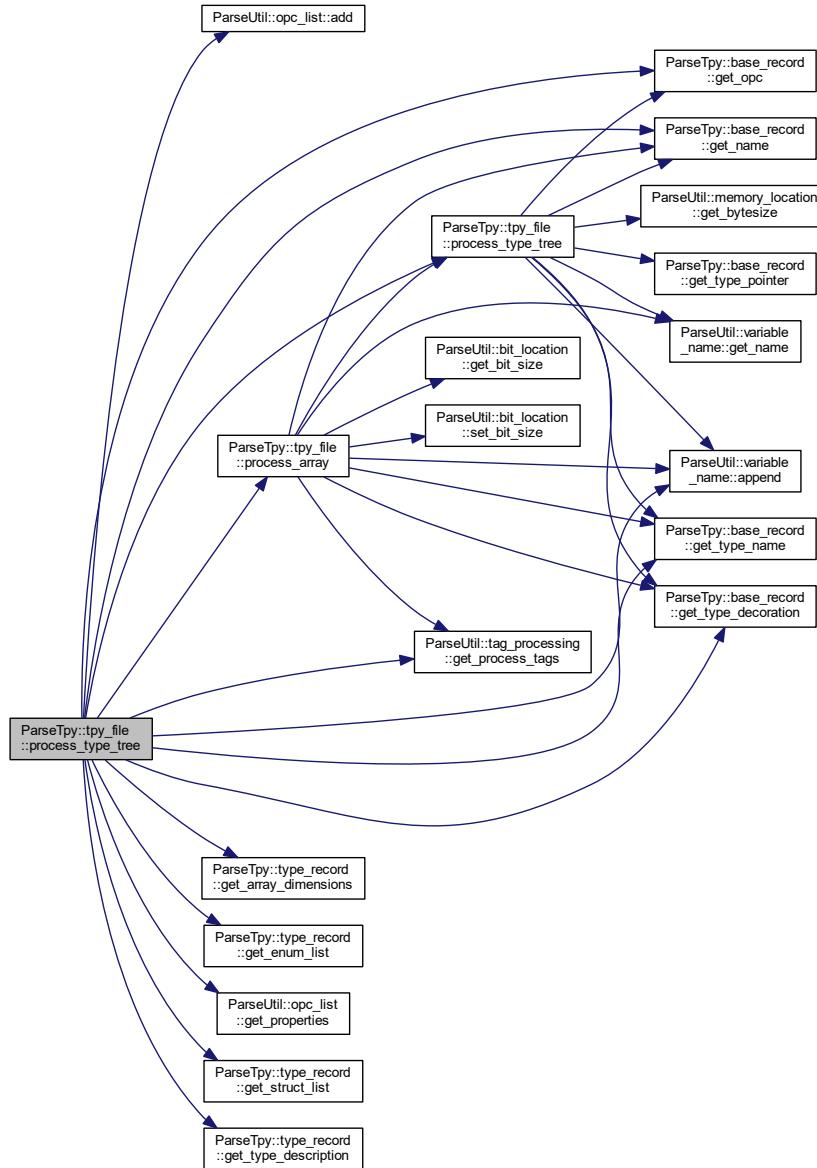
Returns

Number of processes variables

Definition at line 57 of file ParseTpyTemplate.h.

References ParseUtil::opc_list::add(), ParseUtil::variable_name::append(), ParseTpy::arraytype, ParseTpy::enumtype, ParseTpy::functionblock, ParseTpy::type_record::get_array_dimensions(), ParseTpy::type_record::get_enum_list(), ParseTpy::base_record::get_name(), ParseTpy::base_record::get_opc(), ParseUtil::tag_processing::get_process_tags(), ParseUtil::opc_list::get_properties(), ParseTpy::type_record::get_struct_list(), ParseTpy::base_record::get_type_decoration(), ParseTpy::type_record::get_type_description(), ParseTpy::base_record::get_type_name(), ParseUtil::process_all, process_array(), ParseUtil::process_atomic, ParseUtil::process_structured, process_type_tree(), ParseUtil::pt_binary, ParseUtil::pt_enum, ParseUtil::pt_int, ParseTpy::simple, and ParseTpy::structtype.

Here is the call graph for this function:



8.56.2.6 process_type_tree() [3/3]

```

template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const std::string& typ,
    unsigned int id,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
  
```

```
const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
int level = 0 ) const
```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like STRING) is found. Then, calls the process function with an argument of type process_arg.

Parameters

<i>typ</i>	Name of type to resolve
<i>id</i>	Decoration or unique ID of type
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

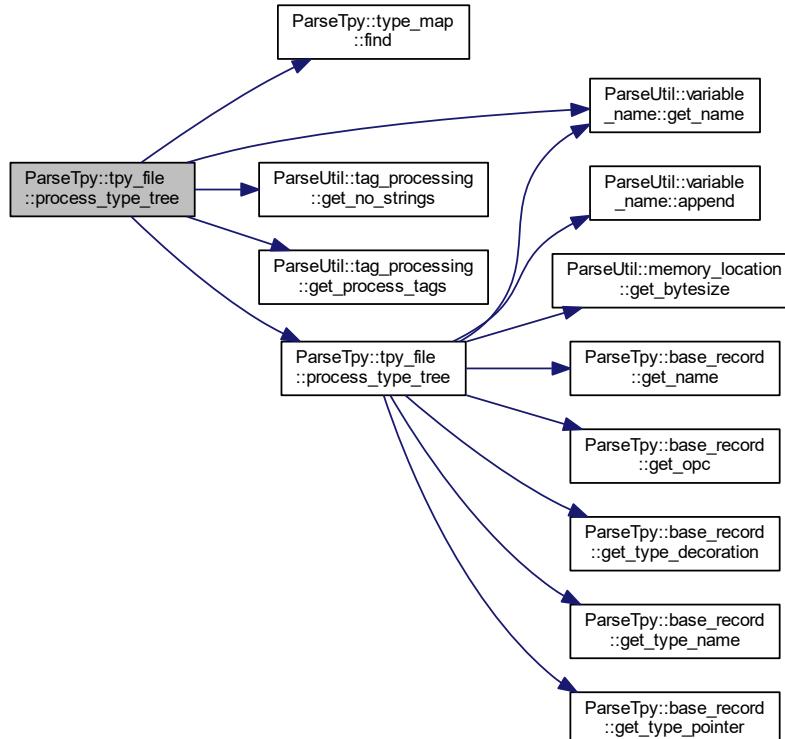
Returns

Number of processes variables

Definition at line 165 of file ParseTpyTemplate.h.

References ParseTpy::type_map::find(), ParseUtil::variable_name::get_name(), ParseUtil::tag_processing::get_no_strings(), ParseUtil::tag_processing::get_process_tags(), ParseUtil::process_all, ParseUtil::process_atomic, process_type_tree(), ParseUtil::pt_bool, ParseUtil::pt_int, ParseUtil::pt_real, ParseUtil::pt_string, and type_list.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)
- [ParseTpyTemplate.h](#)

8.57 DevTc::epics_record_traits< RecType >::traits_type Struct Reference

Epics record type.

```
#include <devTc.h>
```

Public Attributes

- double **val**
Value.

8.57.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::epics_record_traits< RecType >::traits_type
```

Epics record type.

Definition at line 276 of file devTc.h.

The documentation for this struct was generated from the following file:

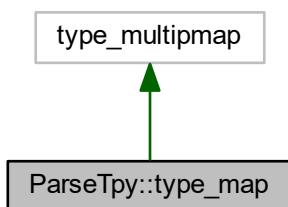
- [devTc.h](#)

8.58 ParseTpy::type_map Class Reference

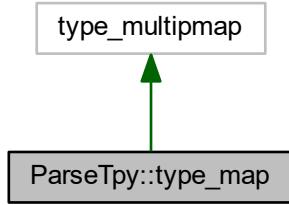
Type dictionary.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type_map:



Collaboration diagram for ParseTpy::type_map:



Public Member Functions

- [type_map \(\)](#)
Constructor.
- [void insert \(value_type val\)](#)
insert a new element
- [const value_type::second_type * find \(value_type::first_type id, const std::stringcase &typn\) const](#)
find an element

8.58.1 Detailed Description

Type dictionary.

This is a map of type records, index is type number as defined in tpy

Definition at line 321 of file ParseTpy.h.

The documentation for this class was generated from the following files:

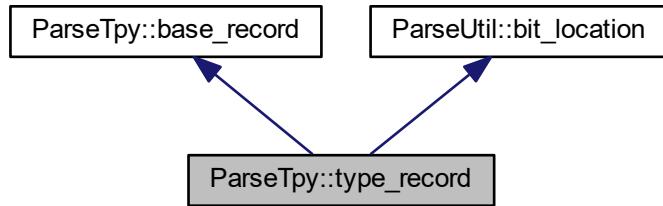
- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.59 ParseTpy::type_record Class Reference

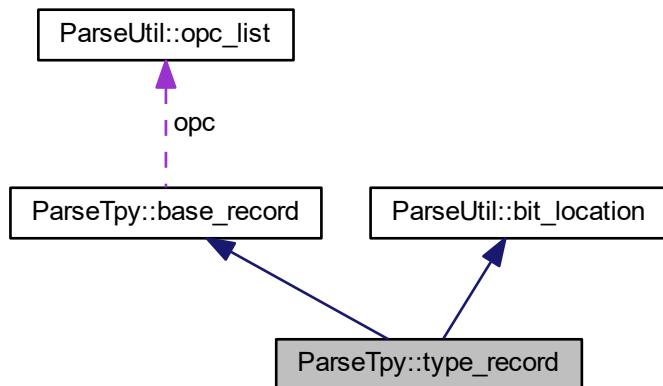
Type record information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type_record:



Collaboration diagram for ParseTpy::type_record:



Public Member Functions

- `type_record ()`
Default constructor.
- `type_enum get_type_description () const`
get the data type
- `void set_type_description (type_enum desc)`
Set decoration (type ID)
- `unsigned int get_name_decoration () const`
Get decoration (type ID)
- `void set_name_decoration (unsigned int id)`
Set decoration (type ID)
- `const dimensions & get_array_dimensions () const`
Get array dimensions.

- [dimensions & get_array_dimensions \(\)](#)
Get array dimensions.
- [const enum_map & get_enum_list \(\) const](#)
Get enumerated list.
- [enum_map & get_enum_list \(\)](#)
Get enumerated list.
- [const item_list & get_struct_list \(\) const](#)
Get structure list.
- [item_list & get_struct_list \(\)](#)
Get structure list.

Protected Attributes

- [type_enum type_desc](#)
Type description.
- [unsigned int name_decoration](#)
decoration (type ID) of type name
- [dimensions array_list](#)
table of dimensions
- [enum_map enum_list](#)
map of enum id and name
- [item_list struct_subitems](#)
list of structure elements

8.59.1 Detailed Description

Type record information.

This structure holds a type record

Definition at line 272 of file ParseTyp.h.

The documentation for this class was generated from the following file:

- [ParseTyp.h](#)

8.60 ParseUtil::variable_name Class Reference

Variable name.

```
#include <ParseUtil.h>
```

Public Member Functions

- [variable_name \(\)](#)
Default constructor.
- [variable_name \(const std::stringcase &n\)](#)
Constructor.
- [variable_name \(const char *s\)](#)
Constructor.
- [variable_name \(const std::stringcase &n, const std::stringcase &a\)](#)
Constructor.
- [const std::stringcase & get_name \(\) const](#)
Get name.
- [const std::stringcase & get_alias \(\) const](#)
Get alias.
- [void set \(const std::stringcase &n\)](#)
Set name & alias.
- [void set \(const std::stringcase &n, const std::stringcase &a\)](#)
Set name & alias.
- [void append \(const std::stringcase &n, const std::stringcase &sep="."\)](#)
Append.
- [void append \(const std::stringcase &n, const opc_list &opc, const std::stringcase &sep="."\)](#)
Append.

Protected Attributes

- [std::stringcase name](#)
variable name
- [std::stringcase alias](#)
alias name

8.60.1 Detailed Description

Variable name.

This is a class for storing a variable name and an alias

Definition at line 192 of file ParseUtil.h.

The documentation for this class was generated from the following files:

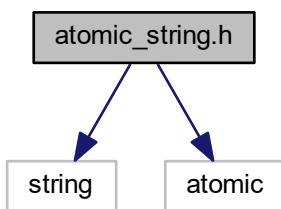
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

Chapter 9

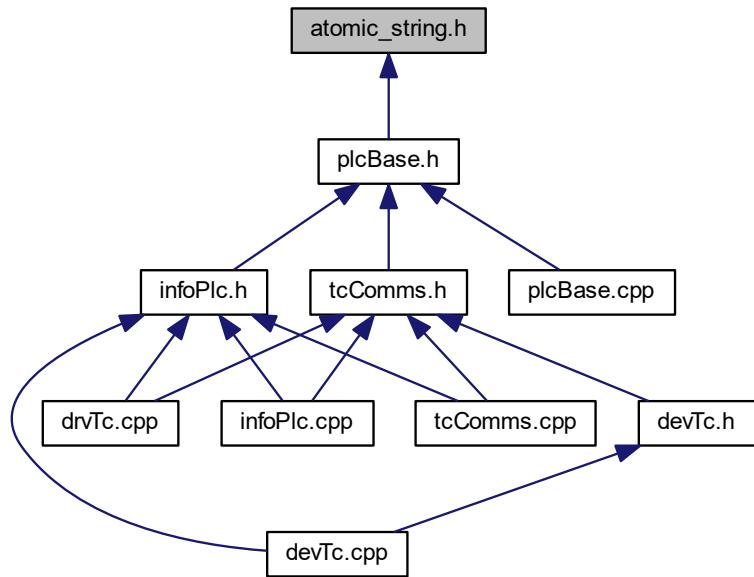
File Documentation

9.1 atomic_string.h File Reference

```
#include <string>
#include <atomic>
Include dependency graph for atomic_string.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `std::atomic< stringT >`
atomic strings
- class `std::atomic< string >`
atomic<string>
- class `std::atomic< wstring >`
atomic<wstring>

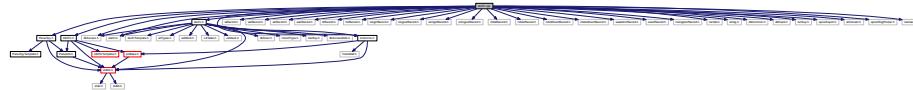
9.1.1 Detailed Description

Header which includes a class for implementing an atomic specialization for string.

9.2 devTc.cpp File Reference

```
#include "devTc.h"
#include "ParseTpy.h"
#include "InfoPlc.h"
#include "errlog.h"
#include "dbAccess.h"
#include "dbCommon.h"
#include "dbEvent.h"
#include "recSup.h"
#include "epicsExport.h"
```

```
#include "aitConvert.h"
#include "epicsRingPointer.h"
#include <iostream>
Include dependency graph for devTc.cpp:
```



Namespaces

- [DevTc](#)
Namespace for TCat device support.

Functions

- [epicsExportAddress \(dset, aival_record_tc_dset\)](#)
Record processing entry for ai.
- [epicsExportAddress \(dset, airval_record_tc_dset\)](#)
Record processing entry for raw ai.
- [epicsExportAddress \(dset, aaival_record_tc_dset\)](#)
Record processing entry for aai.
- [epicsExportAddress \(dset, bival_record_tc_dset\)](#)
Record processing entry for bi.
- [epicsExportAddress \(dset, birval_record_tc_dset\)](#)
Record processing entry for raw bi.
- [epicsExportAddress \(dset, longinval_record_tc_dset\)](#)
Record processing entry for longin.
- [epicsExportAddress \(dset, mbbival_record_tc_dset\)](#)
Record processing entry for mbbi.
- [epicsExportAddress \(dset, mbbirval_record_tc_dset\)](#)
Record processing entry for raw mbbi.
- [epicsExportAddress \(dset, mbbiDirectval_record_tc_dset\)](#)
Record processing entry for mbbiDirect.
- [epicsExportAddress \(dset, mbbiDirectrval_record_tc_dset\)](#)
Record processing entry for raw mbbiDirect.
- [epicsExportAddress \(dset, stringinval_record_tc_dset\)](#)
Record processing entry for stringin.
- [epicsExportAddress \(dset, waveformval_record_tc_dset\)](#)
Record processing entry for waveform.
- [epicsExportAddress \(dset, aoaval_record_tc_dset\)](#)
Record processing entry for ao.
- [epicsExportAddress \(dset, aorval_record_tc_dset\)](#)
Record processing entry for raw ao.
- [epicsExportAddress \(dset, aaoaval_record_tc_dset\)](#)
Record processing entry for aao.
- [epicsExportAddress \(dset, boaval_record_tc_dset\)](#)
Record processing entry for bo.
- [epicsExportAddress \(dset, borval_record_tc_dset\)](#)

- Record processing entry for raw bo.*
- [epicsExportAddress](#) (dset, longoutval_record_tc_dset)

Record processing entry for longout.
 - [epicsExportAddress](#) (dset, mbboval_record_tc_dset)

Record processing entry for mbbo.
 - [epicsExportAddress](#) (dset, mbborval_record_tc_dset)

Record processing entry for raw mbbo.
 - [epicsExportAddress](#) (dset, mbboDirectval_record_tc_dset)

Record processing entry for mbboDirect.
 - [epicsExportAddress](#) (dset, stringoutval_record_tc_dset)

Record processing entry for raw mbboDirect.
 - [epicsExportAddress](#) (dset, stringoutval_record_tc_dset)

Record processing entry for stringout.

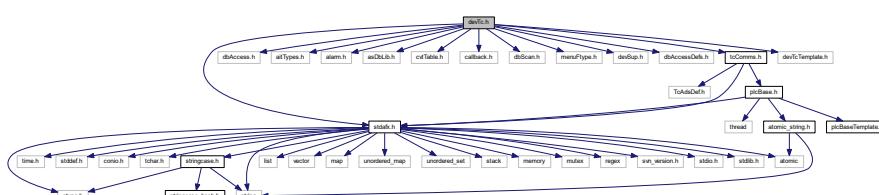
9.2.1 Detailed Description

Methods for TwinCAT/ADS device support.

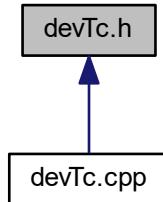
9.3 devTc.h File Reference

```
#include "stdafx.h"
#include "dbAccess.h"
#include "aitTypes.h"
#include "alarm.h"
#include "asDbLib.h"
#include "cvtTable.h"
#include "callback.h"
#include "dbScan.h"
#include "menuFtype.h"
#include "devSup.h"
#include "dbAccessDefs.h"
#include "tcComms.h"
#include "devTcTemplate.h"
```

Include dependency graph for devTc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [DevTc::register_devsup](#)
Device support registration.
- class [DevTc::EpicsInterface](#)
Epics interface class.
- struct [DevTc::epics_record_traits< RecType >](#)
Epics record traits.
- struct [DevTc::epics_record_traits< RecType >::traits_type](#)
Epics record type.
- struct [DevTc::devTcDeflo< RecType >](#)
Device support record.
- struct [DevTc::devTcDefIn< RecType >](#)
Device support input record.
- struct [DevTc::devTcDefOut< RecType >](#)
device support output record.
- struct [DevTc::devTcDefWaveformIn< RecType >](#)
device support waveform record.

Namespaces

- [DevTc](#)
Namespace for TCat device support.

Enumerations

- enum [DevTc::epics_record_enum](#) {
 DevTc::aaival = 0, DevTc::aoval, DevTc::aival, DevTc::aoval,
 DevTc::bival, DevTc::boval, DevTc::eventval, DevTc::histogramval,
 DevTc::longinval, DevTc::longoutval, DevTc::mbbival, DevTc::mbboval,
 DevTc::mbbiDirectval, DevTc::mbboDirectval, DevTc::stringinval, DevTc::stringoutval,
 DevTc::waveformval, DevTc::airval, DevTc::aorval, DevTc::birval,
 DevTc::borval, DevTc::mbbiDirectval, DevTc::mbboDirectval, DevTc::mbbirval,
 DevTc::mbborval, DevTc::epics_record_enumEnd, DevTc::invalidval = -1
 }

Epics record type enum.

Functions

- const std::regex [DevTc::tc_regex](#) ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?)+:([8[0-9][0-9]))|/([\\d{1,9}])/([\\d{1,9}]):([\\d{1,9}])")
Callback for output record.
- const std::regex [DevTc::info_regex](#) ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.|.?)+:([8[0-9][0-9]))|/([info])/([A-Za-z0-9_]+)")
Regex for identifying info records.

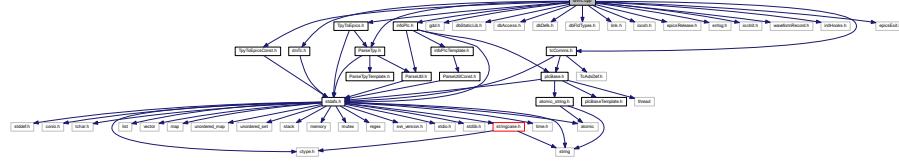
9.3.1 Detailed Description

Header which includes classes for TwinCAT/ADS device support.

9.4 drvTc.cpp File Reference

```
#include "drvTc.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
#include "TpyToEpicsConst.h"
#include "infoPlc.h"
#include "gdd.h"
#include "dbStaticLib.h"
#include "dbAccess.h"
#include "dbDefs.h"
#include "dbFldTypes.h"
#include "link.h"
#include "iocsh.h"
#include "epicsRelease.h"
#include "errlog.h"
#include "iocInit.h"
#include "waveformRecord.h"
#include "initHooks.h"
#include "tcComms.h"
#include "epicsExit.h"
```

Include dependency graph for drvTc.cpp:



Classes

- class [DevTc::epics_tc_db_processing](#)
EPICS/TCat db processing.

Namespaces

- **DevTc**

Namespace for TCat device support.

Typedefs

- **typedef std::tuple< std::stringcase, std::stringcase, epics_list_processing *, bool > DevTc::filename_rule_list_tuple**
Tuple for filnemae, rule and list processing.
- **typedef std::vector< filename_rule_list_tuple > DevTc::tc_listing_def**
List of tuples for filnemae, rule and list processing.
- **typedef std::tuple< std::stringcase, std::stringcase, epics_macrofiles_processing *, const char * > DevTc::dirname_arg_macro_tuple**
Tuple for directory name, argument and macro list processing.
- **typedef std::vector< dirname_arg_macro_tuple > DevTc::tc_macro_def**
List of tuples for directory name, argument and macro list processing.

Functions

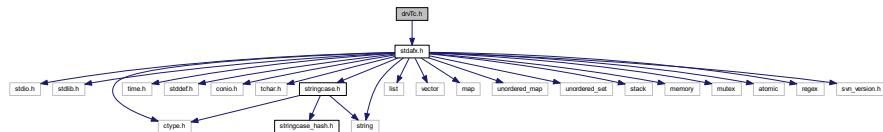
- **void DevTc::tcLoadRecords (const iocshArgBuf *args)**
Load TwinCAT records.
- **void DevTc::tcSetScanRate (const iocshArgBuf *args)**
Set the scan rate.
- **void DevTc::tcList (const iocshArgBuf *args)**
Generate channel lists.
- **void DevTc::tcMacro (const iocshArgBuf *args)**
Generated macro files.
- **void DevTc::tcAlias (const iocshArgBuf *args)**
Define alias and replacement rules.
- **void DevTc::tcInfoPrefix (const iocshArgBuf *args)**
Sets the info prefix.
- **void DevTc::tcPrintVals (const iocshArgBuf *args)**
Print all values.
- **void DevTc::tcPrintVal (const iocshArgBuf *args)**
Print value.

9.4.1 Detailed Description

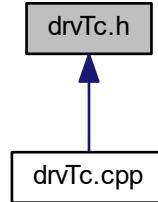
This contains functions for driver support for EPICS. These routines are used during the IOC initialization, and allow the IOC to initialize the TwinCAT interface during EPICS initialization.

9.5 drvTc.h File Reference

```
#include "stdafx.h"
Include dependency graph for drvTc.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [DevTc::tcRegisterToLocShell](#)

Register TC commands.

Namespaces

- [DevTc](#)

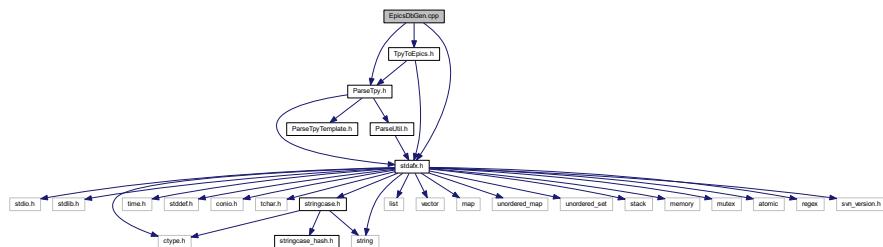
Namespace for TCat device support.

9.5.1 Detailed Description

Header which includes classes for accessing TwinCAT/ADS from EPICS.

9.6 EpicsDbGen.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
Include dependency graph for EpicsDbGen.cpp:
```



Functions

- int main (int argc, char *argv[])

9.6.1 Detailed Description

Source for the main program that generates an EPICs .db file

9.6.2 Function Documentation

9.6.2.1 main()

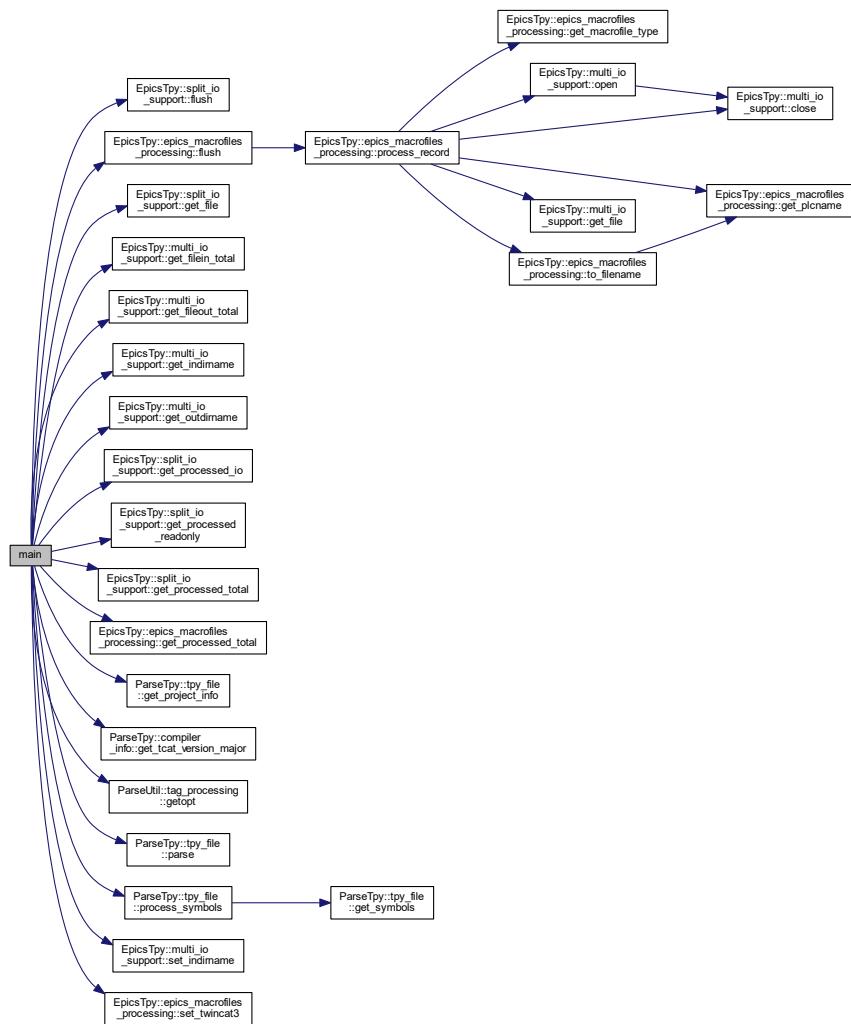
```
int main ( int argc,  
           char * argv[ ] )
```

Main program

Definition at line 18 of file EpicsDbGen.cpp.

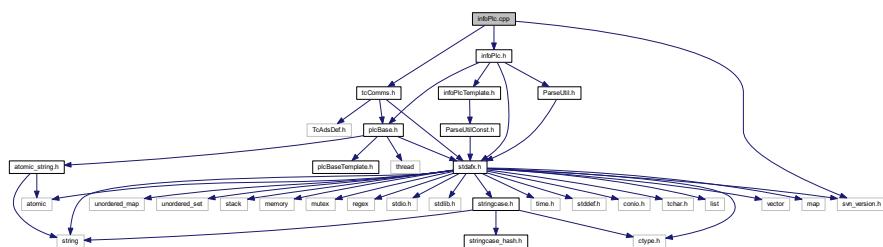
References EpicsTpy::split_io_support::flush(), EpicsTpy::epics_macros_processing::flush(), EpicsTpy::split_io_support::get_file(), EpicsTpy::multi_io_support::get_filein_total(), EpicsTpy::multi_io_support::get_fileout_total(), EpicsTpy::multi_io_support::get_indirname(), EpicsTpy::multi_io_support::get_outdirname(), EpicsTpy::split_io_support::get_processed_io(), EpicsTpy::split_io_support::get_processed_readonly(), EpicsTpy::split_io_support::get_processed_total(), EpicsTpy::epics_macros_processing::get_processed_total(), ParseTpy::tpy_file::get_project_info(), ParseTpy::compiler_info::get_tcat_version_major(), ParseUtil::tag_processing:: getopt(), ParseTpy::tpy_file::parse(), ParseTpy::tpy_file::process_symbols(), EpicsTpy::multi_io_support::set_indirname(), and EpicsTpy::epics_macros_processing::set_twincat3().

Here is the call graph for this function:



9.7 infoPlc.cpp File Reference

```
#include "infoPlc.h"
#include "tcComms.h"
#include "svn_version.h"
Include dependency graph for infoPlc.cpp:
```



Namespaces

- InfoPlc

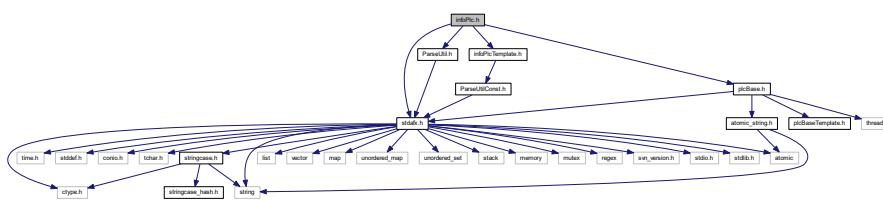
Namespace for Info communication.

9.7.1 Detailed Description

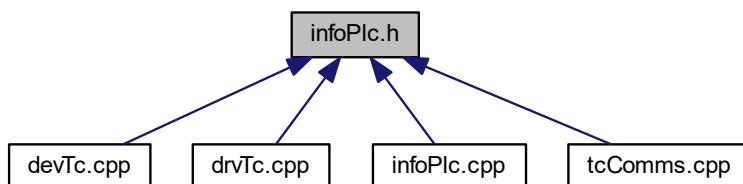
Defines methods for the info PLC.

9.8 infoPlc.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "plcBase.h"
#include "infoPlcTemplate.h"
Include dependency graph for infoPlc.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `InfoPlc::InfoInterface`
Info interface.
 - class `InfoPlc::process_arg_info`
Arguments for processing.

Namespaces

- [InfoPlc](#)

Namespace for Info communication.

Typedefs

- [typedef bool\(InfoInterface::* InfoPlc::info_update_method\) \(\)](#)
Pointer to info update method.
- [typedef std::tuple< ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list, std::stringcase, bool, update_enum, info_update_method > InfoPlc::info_dbrecord_type](#)
- [typedef std::vector< info_dbrecord_type > InfoPlc::info_dbrecord_list](#)
List type of db info tuples.

Enumerations

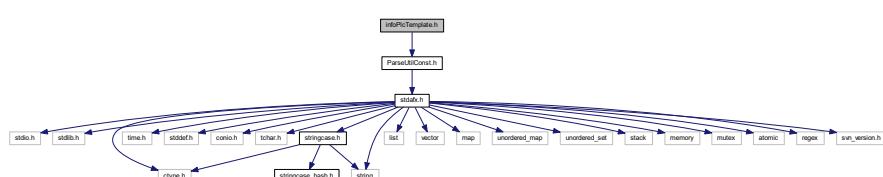
- [enum InfoPlc::update_enum { InfoPlc::update_enum::forever, InfoPlc::update_enum::once, InfoPlc::update_enum::done }](#)
Update frequency type.

9.8.1 Detailed Description

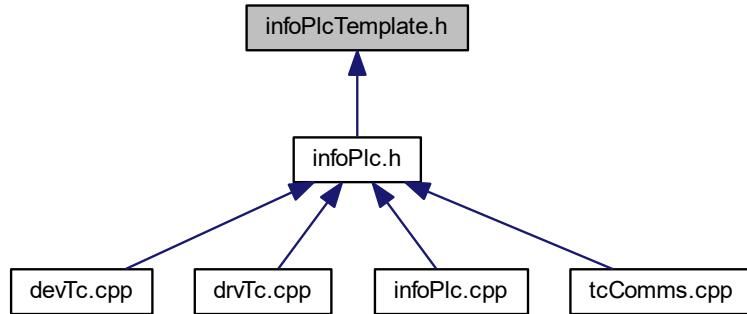
Header which includes classes for the info PLC.

9.9 infoPlcTemplate.h File Reference

```
#include "ParseUtilConst.h"
Include dependency graph for infoPlcTemplate.h:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- [InfoPlc](#)
Namespace for Info communication.

9.9.1 Detailed Description

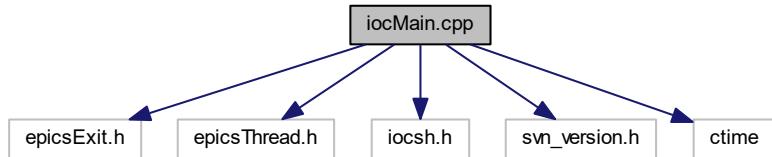
Header which includes classes for the info PLC.

9.10 iocMain.cpp File Reference

```

#include "epicsExit.h"
#include "epicsThread.h"
#include "iocsh.h"
#include "svn_version.h"
#include <ctime>
  
```

Include dependency graph for iocMain.cpp:



Functions

- `__declspec (dllexport) void stopTc(void)`
DLL import for stopTc.
- int `main` (int argc, char *argv[])
`tcloc`

9.10.1 Detailed Description

The main program for the TwinCAT IOC.

9.10.2 Function Documentation

9.10.2.1 main()

```
int main (
    int argc,
    char * argv[ ] )
```

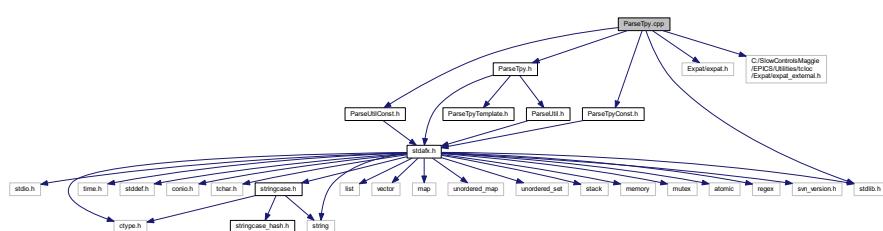
`tcloc`

Main program for `tcloc`

Definition at line 27 of file `iocMain.cpp`.

9.11 ParseTpy.cpp File Reference

```
#include "ParseTpy.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "Expat/expat.h"
Include dependency graph for ParseTpy.cpp:
```



Classes

- class `ParseTpy::parserinfo_type`
Parser information.

Namespaces

- `ParseTpy`

Namespace for parsing.

Macros

- `#define XML_STATIC`
Static linking.
- `#define XML_FMT_INT_MOD "I"`
Int format.

Functions

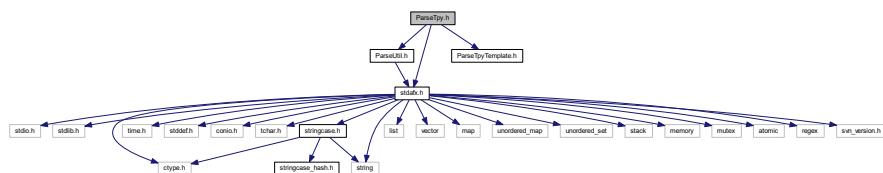
- `bool ParseTpy::compareNamesWoNamespace (const std::stringcase &p1, const std::stringcase &p2)`
- `bool ParseTpy::get_decoration (const char **atts, unsigned int &decoration)`
- `bool ParseTpy::get_pointer (const char **atts)`

9.11.1 Detailed Description

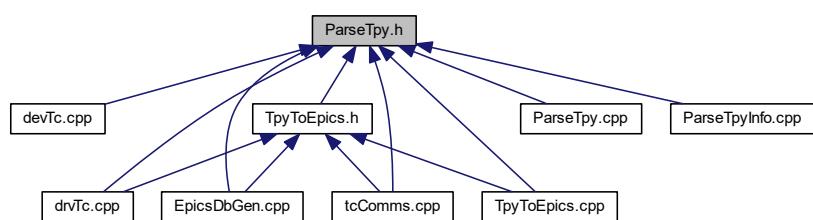
Methods to parse TwinCAT tpy file.

9.12 ParseTpy.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "ParseTpyTemplate.h"
Include dependency graph for ParseTpy.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ParseTpy::ads_routing_info](#)
ADS routing information.
- class [ParseTpy::compiler_info](#)
Compiler information.
- class [ParseTpy::project_record](#)
Project information.
- class [ParseTpy::base_record](#)
Base record definition.
- class [ParseTpy::item_record](#)
item record
- class [ParseTpy::type_record](#)
Type record information.
- class [ParseTpy::type_map](#)
Type dictionary.
- class [ParseTpy::symbol_record](#)
Symbol record.
- class [ParseTpy::tpy_file](#)
Tpy file parsing.

Namespaces

- [ParseTpy](#)
Namespace for parsing.

Typedefs

- `typedef std::pair< int, int > ParseTpy::dimension`
- `typedef std::list< dimension > ParseTpy::dimensions`
- `typedef std::map< int, std::stringcase > ParseTpy::enum_map`
- `typedef std::pair< int, std::stringcase > ParseTpy::enum_pair`
- `typedef std::list< item_record > ParseTpy::item_list`
- `typedef std::multimap< unsigned int, type_record > ParseTpy::type_multipmap`
- `typedef std::list< symbol_record > ParseTpy::symbol_list`

Enumerations

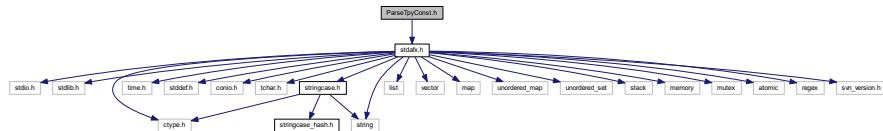
- `enum ParseTpy::type_enum {
 ParseTpy::unknown, ParseTpy::simple, ParseTpy::arraytype, ParseTpy::enumtype,
 ParseTpy::structtype, ParseTpy::functionblock }`
Type enum.

9.12.1 Detailed Description

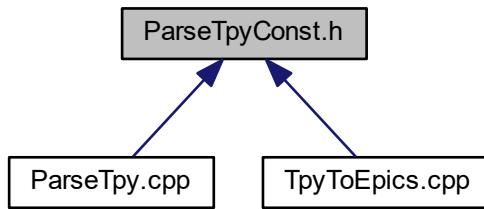
Header which includes classes to parse a TwinCAT tpy file. It includes "ParseUtil.h" and "ParseTpyTemplate.h"

9.13 ParseTpyConst.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseTpyConst.h:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseUtil](#)
Namespace for parsing utilities.
- [ParseTpy](#)
Namespace for parsing.

Variables

- `const char *const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"`
PLC project info.
- `const char *const ParseTpy::xmlProjectInfo = "ProjectInfo"`
Project info.
- `const char *const ParseTpy::xmlRoutingInfo = "RoutingInfo"`
Routing info.
- `const char *const ParseTpy::xmlCompilerInfo = "CompilerInfo"`
Compiler info.
- `const char *const ParseTpy::xmlAdsInfo = "AdsInfo"`
ADS info.
- `const char *const ParseTpy::xmlDataTypes = "DataTypes"`
Data types.
- `const char *const ParseTpy::xmlDataType = "DataType"`

- const char *const ParseTpy::xmlSymbols = "Symbols"
Symbols.
- const char *const ParseTpy::xmlSymbol = "Symbol"
Symbol.
- const char *const ParseTpy::xmlProperties = "Properties"
Properties.
- const char *const ParseTpy::xmlProperty = "Property"
Property.
- const char *const ParseTpy::xmlCompilerVersion = "CompilerVersion"
Compiler version.
- const char *const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"
TwinCAT version.
- const char *const ParseTpy::xmlCpuFamily = "CpuFamily"
CPU family.
- const char *const ParseTpy::xmlNetId = "NetId"
Net ID.
- const char *const ParseTpy::xmlPort = "Port"
Port.
- const char *const ParseTpy::xmlTargetName = "TargetName"
Target name.
- const char *const ParseTpy::xmlName = "Name"
Name.
- const char *const ParseTpy::xmlType = "Type"
Type.
- const char *const ParseTpy::xmlAttrDecoration = "Decoration"
Decoration.
- const char *const ParseTpy::xmlAttrPointer = "Pointer"
Pointer.
- const char *const ParseTpy::xmlIGroup = "IGroup"
I Group.
- const char *const ParseTpy::xmlIOffset = "IOffset"
I Offset.
- const char *const ParseTpy::xmlBitSize = "BitSize"
Bit size.
- const char *const ParseTpy::xmlBitOffs = "BitOffs"
Bit Offset.
- const char *const ParseTpy::xmlArrayInfo = "ArrayInfo"
Array info.
- const char *const ParseTpy::xmlArrayLBound = "LBound"
Lower bound.
- const char *const ParseTpy::xmlArrayElements = "Elements"
Elements.
- const char *const ParseTpy::xmlSubItem = "SubItem"
Sub item.
- const char *const ParseTpy::xmlFbInfo = "FbInfo"
Fb info.
- const char *const ParseTpy::xmlEnumInfo = "EnumInfo"
Enum info.
- const char *const ParseTpy::xmlEnumText = "Text"
Text.

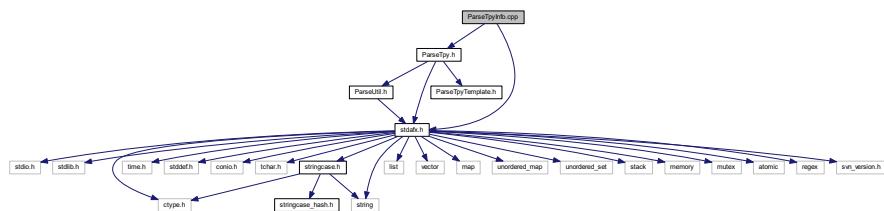
- const char *const ParseTpy::xmlEnumEnum = "Enum"
Enum.
- const char *const ParseTpy::xmlEnumComment = "Comment"
Comment.
- const char *const ParseTpy::xmlValue = "Value"
Value.
- const char *const ParseTpy::xmlDesc = "Desc"
Description.
- const char *const ParseTpy::opcExport = "opc"
OPC.
- const char *const ParseTpy::opcProp = "opc_prop"
OPC property.
- const char *const ParseTpy::opcBracket = "["
OPC bracket.

9.13.1 Detailed Description

Header which includes constants needed to parse a TwinCAT tpy file.

9.14 ParseTpyInfo.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
Include dependency graph for ParseTpyInfo.cpp:
```



Classes

- class **syminfo_processing**
Symbol processing.

Functions

- int **main** (int argc, char *argv[])

tpyinfo

9.14.1 Detailed Description

Source file for tpy parsing methods.

9.14.2 Function Documentation

9.14.2.1 main()

```
int main (
    int argc,
    char * argv[] )
```

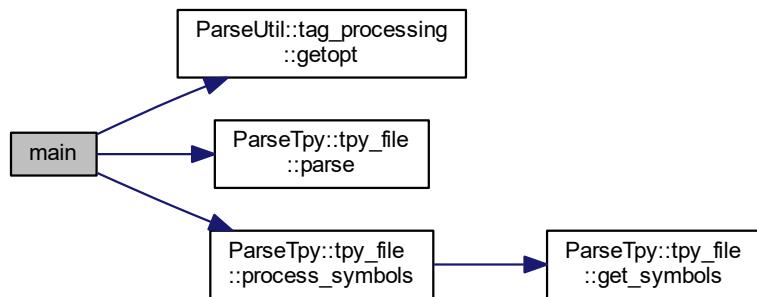
tpyinfo

Main program for tpyinfo

Definition at line 94 of file ParseTpyInfo.cpp.

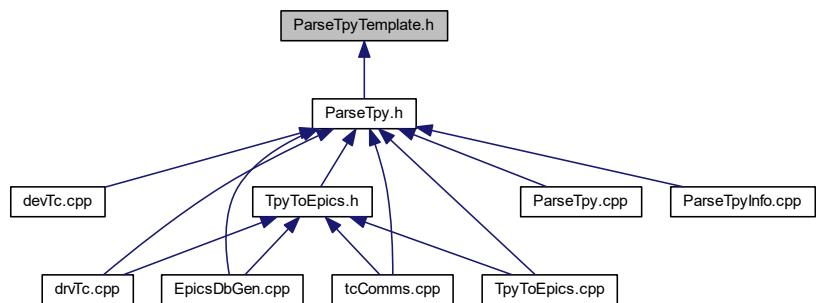
References ParseUtil::tag_processing::getopt(), ParseTpy::tpy_file::parse(), and ParseTpy::tpy_file::process_symbols().

Here is the call graph for this function:



9.15 ParseTpyTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseTpy](#)

Namespace for parsing.

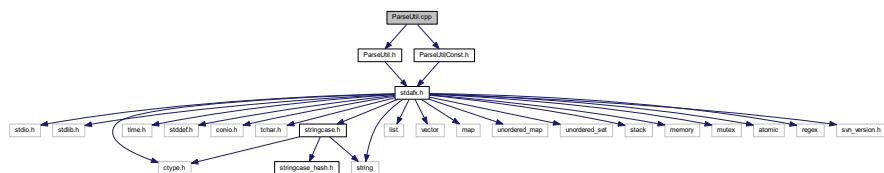
9.15.1 Detailed Description

Templates for functions for parsing tpy file.

9.16 ParseUtil.cpp File Reference

```
#include "ParseUtil.h"
#include "ParseUtilConst.h"
```

Include dependency graph for ParseUtil.cpp:



Namespaces

- [ParseUtil](#)

Namespace for parsing utilities.

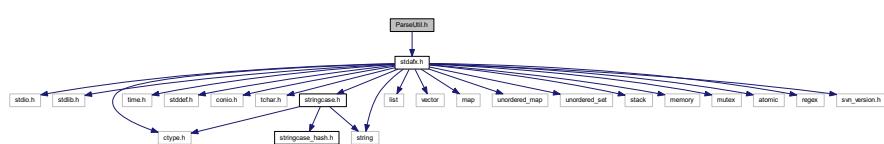
9.16.1 Detailed Description

Utility methods to parsing.

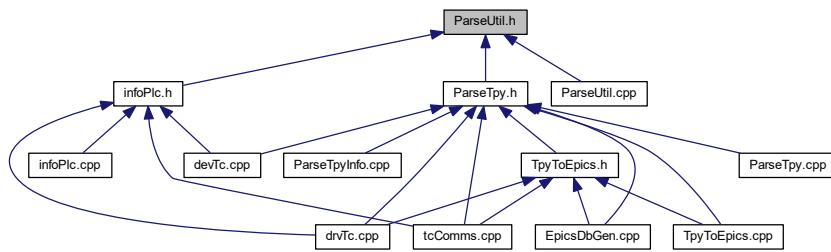
9.17 ParseUtil.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for ParseUtil.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ParseUtil::replacement_rules](#)
Replacement rules.
- class [ParseUtil::optarg](#)
Optional arguments.
- class [ParseUtil::opc_list](#)
OPC list.
- class [ParseUtil::variable_name](#)
Variable name.
- class [ParseUtil::bit_location](#)
Bit location.
- class [ParseUtil::memory_location](#)
Memory location.
- class [ParseUtil::process_arg](#)
Arguments for processing.
- class [ParseUtil::process_arg_tc](#)
Arguments for processing.
- class [ParseUtil::tag_processing](#)
Tag processing selection.

Namespaces

- [ParseUtil](#)
Namespace for parsing utilities.

TypeDefs

- [typedef std::map< std::stringcase, std::stringcase > ParseUtil::replacement_table](#)
- [typedef std::map< int, std::stringcase > ParseUtil::property_map](#)
- [typedef std::pair< int, std::stringcase > ParseUtil::property_el](#)

Enumerations

- enum `ParseUtil::opc_enum` { `ParseUtil::no_change`, `ParseUtil::publish`, `ParseUtil::silent` }
OPC state enum.
- enum `ParseUtil::process_type_enum` {
`ParseUtil::pt_invalid`, `ParseUtil::pt_int`, `ParseUtil::pt_real`, `ParseUtil::pt_bool`,
`ParseUtil::pt_string`, `ParseUtil::pt_enum`, `ParseUtil::pt_binary` }
Process type.
- enum `ParseUtil::process_tag_enum` { `ParseUtil::process_all`, `ParseUtil::process_atomic`, `ParseUtil::process_structured` }
Tag preoicing enum.

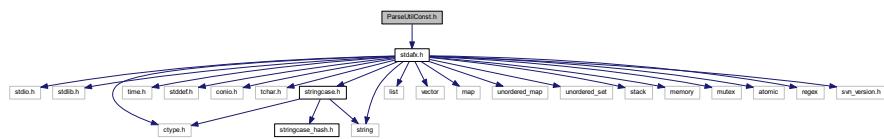
9.17.1 Detailed Description

Header which includes utility classes for parsing. It includes "ParseUtilTemplate.h"

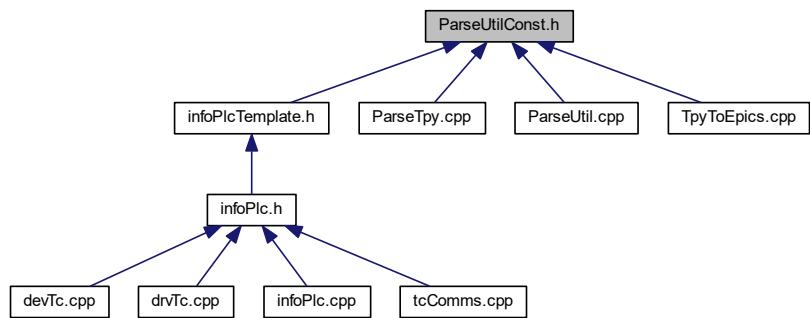
9.18 ParseUtilConst.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for ParseUtilConst.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- `ParseTpy`
Namespace for parsing.
- `ParseUtil`
Namespace for parsing utilities.

Variables

- const int ParseUtil::OPC_PROP_CDT = 1
- const int ParseUtil::OPC_PROP_VALUE = 2
- const int ParseUtil::OPC_PROP_QUALITY = 3
- const int ParseUtil::OPC_PROP_TIME = 4
- const int ParseUtil::OPC_PROP_RIGHTS = 5
- const int ParseUtil::OPC_PROP_SCANRATE = 6
- const int ParseUtil::OPC_PROP_UNIT = 100
- const int ParseUtil::OPC_PROP_DESC = 101
- const int ParseUtil::OPC_PROP_HIEU = 102
- const int ParseUtil::OPC_PROP_LOEU = 103
- const int ParseUtil::OPC_PROP_HIRANGE = 104
- const int ParseUtil::OPC_PROP_LORANGE = 105
- const int ParseUtil::OPC_PROP_CLOSE = 106
- const int ParseUtil::OPC_PROP_OPEN = 107
- const int ParseUtil::OPC_PROP_TIMEZONE = 108
- const int ParseUtil::OPC_PROP_FGC = 201
- const int ParseUtil::OPC_PROP_BGC = 202
- const int ParseUtil::OPC_PROP_BLINK = 203
- const int ParseUtil::OPC_PROP_BMP = 204
- const int ParseUtil::OPC_PROP SND = 205
- const int ParseUtil::OPC_PROP_HTML = 206
- const int ParseUtil::OPC_PROP_AVI = 207
- const int ParseUtil::OPC_PROP_ALMSTAT = 300
- const int ParseUtil::OPC_PROP_ALMHELP = 301
- const int ParseUtil::OPC_PROP_ALMAREAS = 302
- const int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303
- const int ParseUtil::OPC_PROP_ALMCONDITION = 304
- const int ParseUtil::OPC_PROP_ALMLIMIT = 305
- const int ParseUtil::OPC_PROP_ALMDB = 306
- const int ParseUtil::OPC_PROP_ALMH = 307
- const int ParseUtil::OPC_PROP_ALMH = 308
- const int ParseUtil::OPC_PROP_ALML = 309
- const int ParseUtil::OPC_PROP_ALMLL = 310
- const int ParseUtil::OPC_PROP_ALMROC = 311
- const int ParseUtil::OPC_PROP_ALMDEV = 312
- const int ParseUtil::OPC_PROP_PREC = 8500
- const int ParseUtil::OPC_PROP_ZRST = 8510
- const int ParseUtil::OPC_PROP_FFST = 8525
- const int ParseUtil::OPC_PROP_RECTYPE = 8600
- const int ParseUtil::OPC_PROP_INOUT = 8601
- const char *const ParseUtil::OPC_PROP_INPUT = "input"
- const char *const ParseUtil::OPC_PROP_OUTPUT = "output"
- const int ParseUtil::OPC_PROP_TSE = 8602
- const int ParseUtil::OPC_PROP_PINI = 8603
- const int ParseUtil::OPC_PROP_DTYP = 8604
- const int ParseUtil::OPC_PROP_SERVER = 8610
- const int ParseUtil::OPC_PROP_PLNAME = 8611
- const int ParseUtil::OPC_PROP_ALIAS = 8620
- const int ParseUtil::OPC_PROP_ALMOSV = 8700
- const int ParseUtil::OPC_PROP_ALMZSV = 8701
- const int ParseUtil::OPC_PROP_ALMCOSV = 8702
- const int ParseUtil::OPC_PROP_ALMUNSV = 8703
- const int ParseUtil::OPC_PROP_ALMZRSV = 8710

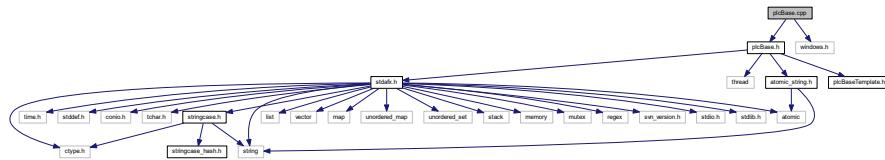
- const int ParseUtil::OPC_PROP_ALMFFSV = 8725
- const int ParseUtil::OPC_PROP_ALMHHSV = 8727
- const int ParseUtil::OPC_PROP_ALMHSV = 8728
- const int ParseUtil::OPC_PROP_ALMLSV = 8729
- const int ParseUtil::OPC_PROP_ALMLLSV = 8730

9.18.1 Detailed Description

Header which includes OPC constants.

9.19 plcBase.cpp File Reference

```
#include "plcBase.h"
#include <windows.h>
Include dependency graph for plcBase.cpp:
```



Classes

- struct `plc::scanner_thread_args`
- Scanner thread arguments.*

Namespaces

- `plc`
- Namespace for abstract plc functionality.*

Functions

- template<>
bool `plc::reset_and_read` (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, DataValueTypeDef::atomic_string *source)

Reset and read.

- template<>
bool `plc::write_and_test` (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source)

Write and test.

- VOID CALLBACK `plc::ScannerProc` (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)

Scanner thread callback.

- DWORD WINAPI `plc::scannerThread` (scanner_thread_args args)

Scanner thread.

- `__declspec(dllexport) void stopTc(void)`

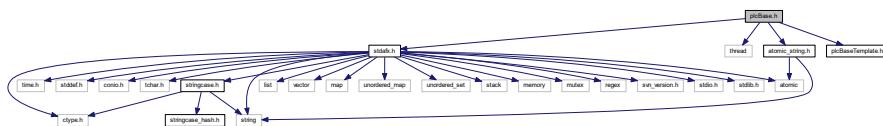
Stop TwinCAT.

9.19.1 Detailed Description

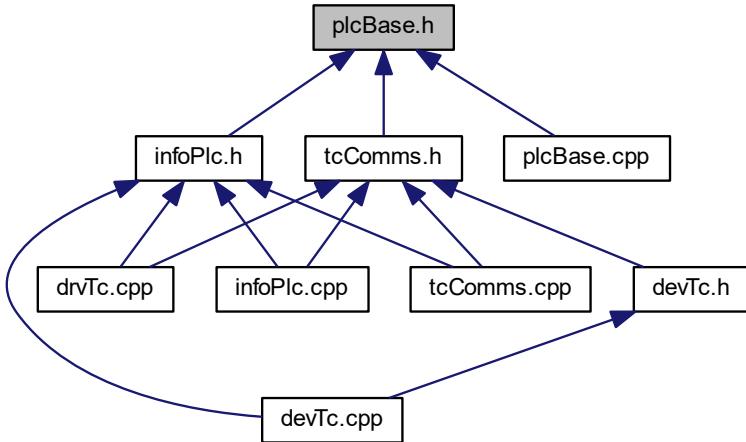
Defines methods for the internal record entry.

9.20 plcBase.h File Reference

```
#include "stdafx.h"
#include <thread>
#include "atomic_string.h"
#include "plcBaseTemplate.h"
Include dependency graph for plcBase.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [plc::Interface](#)
Abstract interface.
- struct [plc::DataValueTraits< T >](#)
Data value traits.
- struct [plc::DataValueTypeDef](#)
Collection of type definitions.
- class [plc::DataValue](#)

Data value.

- class [plc::BaseRecord](#)
Class for managing a tag/channel.
- class [plc::BasePLC](#)
Base PLC.
- class [plc::System](#)
System to keep track of PLCs.

Namespaces

- [plc](#)
Namespace for abstract plc functionality.

Typedefs

- [typedef std::shared_ptr< BasePLC > plc::BasePLCPtr](#)
Smart pointer to PLC.
- [typedef std::unique_ptr< Interface > plc::InterfacePtr](#)
Smart pointer to interface.
- [typedef std::shared_ptr< BaseRecord > plc::BaseRecordPtr](#)
smart pointer to record
- [typedef std::unordered_map< std::stringcase, BaseRecordPtr > plc::BaseRecordList](#)
list of record
- [typedef std::map< std::stringcase, BasePLCPtr > plc::BasePLCList](#)
BasePLC map.

Enumerations

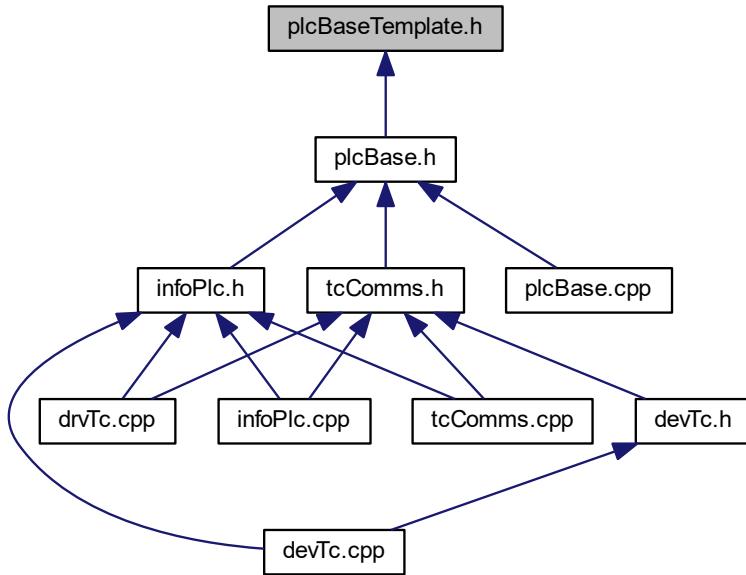
- enum [plc::data_type_enum](#) {
 [plc::dtInvalid](#), [plc::dtBool](#), [plc::dtInt8](#), [plc::dtUInt8](#),
 [plc::dtInt16](#), [plc::dtUInt16](#), [plc::dtInt32](#), [plc::dtUInt32](#),
 [plc::dtInt64](#), [plc::dtUInt64](#), [plc::dtFloat](#), [plc::dtDouble](#),
 [plc::dtString](#), [plc::dtWString](#), [plc::dtBinary](#) }
- enum [plc::access_rights_enum](#) { [plc::read_only](#), [plc::write_only](#), [plc::read_write](#) }
Access rights enum.

9.20.1 Detailed Description

Header which includes abstract base classes for defining an internal record entry for the IOC.

9.21 plcBaseTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- `plc`

Namespace for abstract plc functionality.

Functions

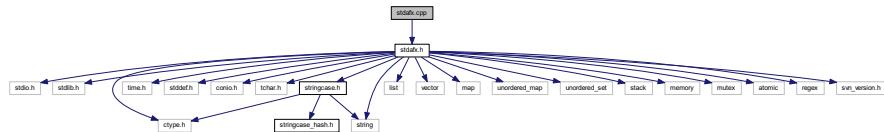
- template<typename T , typename U >
`bool plc::reset_and_read` (DataValueTypeDef::atomic_bool &dirty, T &dest, U source)
Reset and read.
- template<typename T >
`bool plc::reset_and_read` (DataValueTypeDef::atomic_bool &dirty, T &dest, typename DataValueTraits< T >::traits_atomic *source)
Reset and read.
- template<typename T , typename U >
`bool plc::write_and_test` (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, U dest, const T &source)
Write and test.
- template<typename T >
`bool plc::write_and_test` (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, typename DataValueTraits< T >::traits_atomic *dest, const T &source)
Write and test.

9.21.1 Detailed Description

Header which includes templated methods for abstract record class and the DataValue class.

9.22 stdafx.cpp File Reference

```
#include "stdafx.h"
Include dependency graph for stdafx.cpp:
```

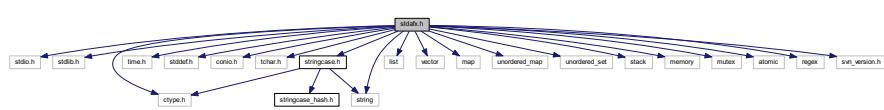


9.22.1 Detailed Description

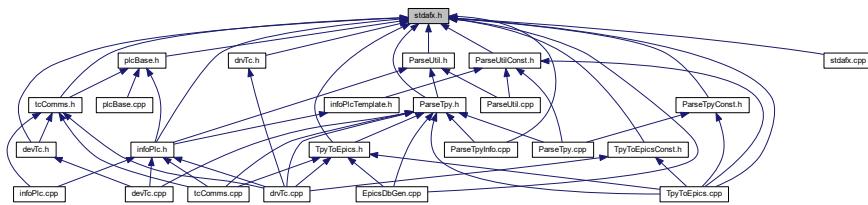
source file that includes just the standard includes EpicsDbGen.pch will be the pre-compiled header stdafx.obj will contain the pre-compiled type information

9.23 stdafx.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <stddef.h>
#include <conio.h>
#include <tchar.h>
#include <string>
#include "stringcase.h"
#include <list>
#include <vector>
#include <map>
#include <unordered_map>
#include <unordered_set>
#include <stack>
#include <memory>
#include <mutex>
#include <atomic>
#include <regex>
#include "svn_version.h"
Include dependency graph for stdafx.h:
```



This graph shows which files directly or indirectly include this file:

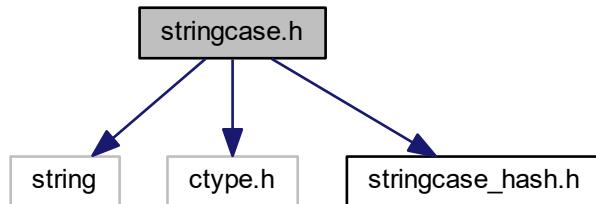


9.23.1 Detailed Description

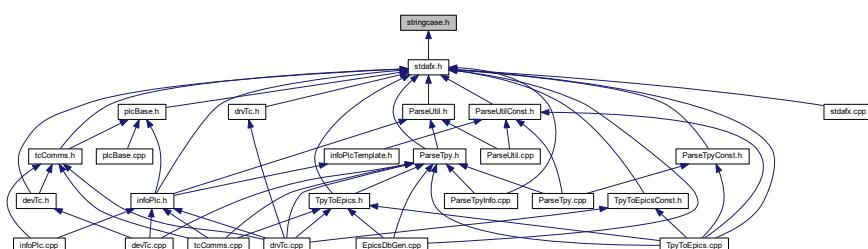
Include file for standard system include files, or project specific include files that are used frequently, but are changed infrequently

9.24 stringcase.h File Reference

```
#include <string>
#include <ctype.h>
#include "stringcase_hash.h"
Include dependency graph for stringcase.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct `std::case_char_traits`
`case insensitive traits.`
- struct `std::case_wchar_traits`
`case insensitive unicode traits.`

TypeDefs

- typedef `std::basic_string< char, case_char_traits > std::stringcase`
`case insensitive string.`
- typedef `std::basic_string< wchar_t, case_wchar_traits > std::wstringcase`
`case insensitive string.`

Functions

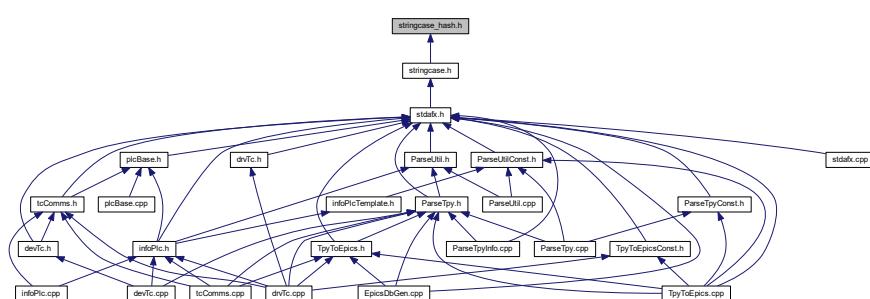
- int `std::strncasecmp` (const char *s1, const char *s2, size_t n)
`case insensitive compare with maximum length`
- int `std::wcsncasewcncmp` (const wchar_t *s1, const wchar_t *s2, size_t n)
`case insensitive unicode compare with maximum length`
- void `std::trim_space` (`std::stringcase &s`)
- void `std::trim_space` (`std::wstringcase &s`)
- template<class Container , class String , class Predicate >
`void std::split_string` (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)
Splits a strings.

9.24.1 Detailed Description

Functions and classes for a case-insensitive string.

9.25 stringcase_hash.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct `std::std::hash< std::stringcase >`
hash for case insensitive string.
- struct `std::std::hash< std::wstringcase >`
hash for case insensitive unicode string.

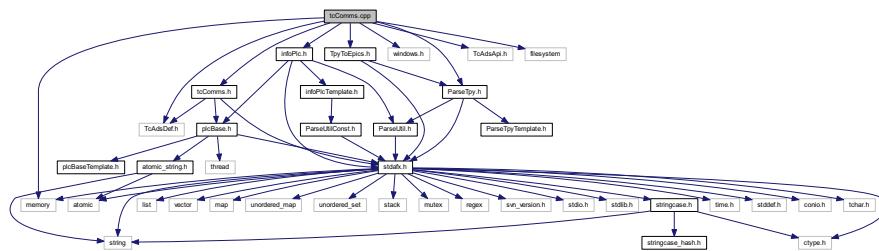
9.25.1 Detailed Description

Specialization for case sensitive strings.

9.26 tcComms.cpp File Reference

```
#include "tcComms.h"
#include "infoPlc.h"
#include "ParseTpy.h"
#include "windows.h"
#include "TpyToEpics.h"
#include "TcAdsDef.h"
#include "TcAdsApi.h"
#include <memory>
#include <filesystem>
```

Include dependency graph for tcComms.cpp:



Namespaces

- **TcComms**
Namespace for TCat communication.

Functions

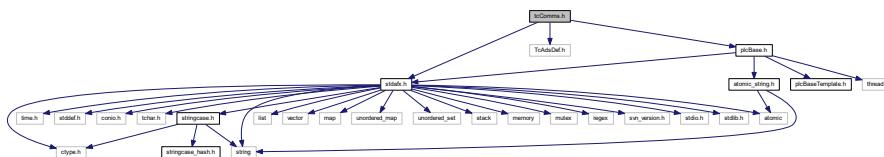
- void `TcComms::errorPrintf (int nErr)`
`errorPrintf`
- void `__stdcall TcComms::ADScallback (AmsAddr *pAddr, AdsNotificationHeader *pNotification, unsigned long plcid)`
- void `__stdcall TcComms::RouterCall (long nReason)`

9.26.1 Detailed Description

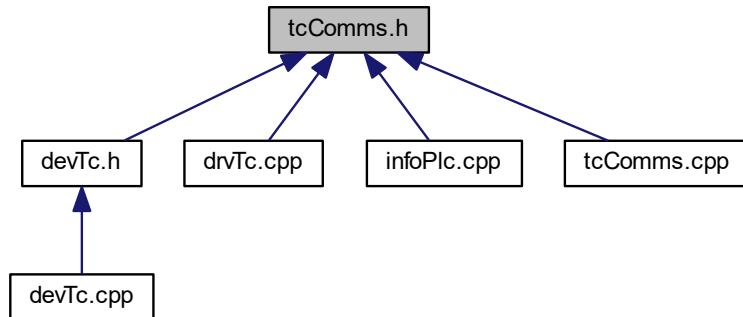
Defines methods for TwinCAT communication.

9.27 tcComms.h File Reference

```
#include "stdafx.h"
#include "TcAdsDef.h"
#include "plcBase.h"
Include dependency graph for tcComms.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct **TcComms::DataPar**
Memory location struct.
 - class **TcComms::TCatInterface**
TCat interface class.
 - class **TcComms::tcProcWrite**
TwinCAT process write requests.
 - class **TcComms::TcPLC**
TwinCAT PLC.
 - class **TcComms::AmsRouterNotification**
AMS Router Notification.

Namespaces

- **TcComms**

Namespace for TCat communication.

Variables

- const int **TcComms::MAX_REQ_SIZE** = 250000
maximum allowed request size (bytes)
- const int **TcComms::MAX_SINGLE_GAP_SIZE** = 50
maximum allowed size (bytes) of a memory gap within continuous request
- const double **TcComms::MAX_REL_GAP** = 0.25
(maximum allowed total gap size) / (current request size)
- const int **TcComms::MIN_REL_GAP_SIZE** = 100
minimum allowed relative gap size (bytes)
- const int **TcComms::default_scanrate** = 100
default PLC TwinCAT scan rate (100ms)
- const int **TcComms::minimum_scanrate** = 5
minimum PLC TwinCAT scan rate (5ms)
- const int **TcComms::maximum_scanrate** = 10000
maximum PLC TwinCAT scan rate (10s)
- const int **TcComms::default_multiple** = 10
default multiple for PLC EPICS scan rate (10)
- const int **TcComms::minimum_multiple** = 1
minimum multiple for PLC EPICS scan rate (1)
- const int **TcComms::maximum_multiple** = 200
maximum multiple for PLC EPICS scan rate (200)

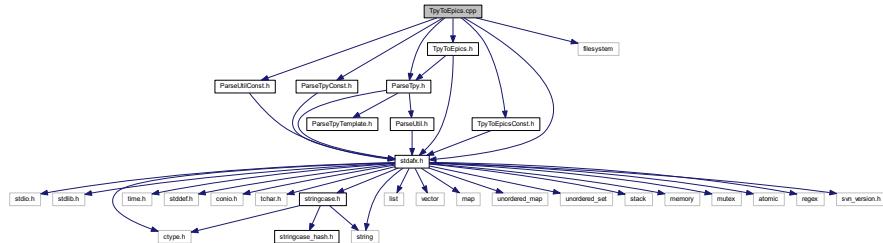
9.27.1 Detailed Description

Header which includes classes to interface with the TCat system and manage TCat symbols.

9.28 TpyToEpics.cpp File Reference

```
#include "stdafx.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "ParseTpy.h"
#include "TpyToEpicsConst.h"
#include "TpyToEpics.h"
#include <filesystem>
```

Include dependency graph for TpyToEpics.cpp:



Namespaces

- EpicsTpy

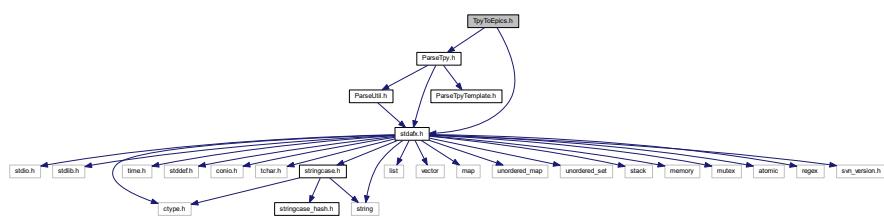
Namespace for tpy-db conversion.

9.28.1 Detailed Description

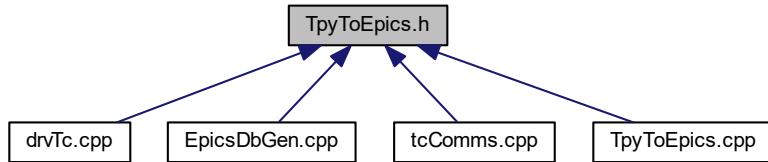
Source for methods that generate EPICs .db file from a .tpy file

9.29 TpyToEpics.h File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
Include dependency graph for TpyToEpics.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `EpicsTpy::epics_conversion`
Epics conversion.
 - class `EpicsTpy::split_io_support`
Split IO support.
 - class `EpicsTpy::multi_io_support`
Multiple IO support.
 - class `EpicsTpy::epics_list_processing`
List processing.
 - struct `EpicsTpy::macro_info`

- struct [EpicsTpy::macro_record](#)

Macro record.
- class [EpicsTpy::epics_macrofiles_processing](#)

Macro file processing.
- class [EpicsTpy::epics_db_processing](#)

pics database record processing

Namespaces

- [EpicsTpy](#)

Namespace for tpy-db conversion.

Typedefs

- typedef std::vector< macro_info > [EpicsTpy::macro_list](#)
- typedef std::stack< macro_record > [EpicsTpy::macro_stack](#)
- typedef std::unordered_set< std::stringcase > [EpicsTpy::filename_set](#)

Enumerations

- enum [EpicsTpy::tc_epics_conv](#) { [EpicsTpy::no_conversion](#), [EpicsTpy::no_dot](#), [EpicsTpy::ligo_std](#), [EpicsTpy::ligo_vac](#) }

Conversion rules for TC/EPICS.
- enum [EpicsTpy::case_type](#) { [EpicsTpy::preserve_case](#), [EpicsTpy::upper_case](#), [EpicsTpy::lower_case](#) }

Case conversion rule enum.
- enum [EpicsTpy::io_filestat](#) { [EpicsTpy::io_filestat::closed](#), [read](#), [write](#) }

enum for file io
- enum [EpicsTpy::listing_type](#) { [EpicsTpy::listing_standard](#), [EpicsTpy::listing_autoburt](#), [EpicsTpy::listing_daqini](#) }

Listing type enum.
- enum [EpicsTpy::macrofile_type](#) { [EpicsTpy::macrofile_type::all](#), [EpicsTpy::macrofile_type::fields](#), [EpicsTpy::macrofile_type::error](#) }
- enum [EpicsTpy::device_support_type](#) { [EpicsTpy::device_support_opc_name](#), [EpicsTpy::device_support_tc_name](#) }

Device support enum.

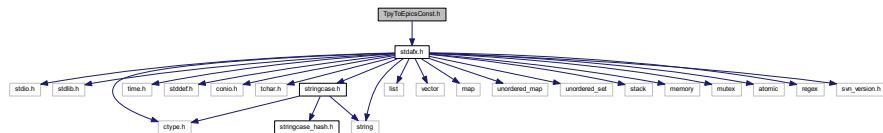
9.29.1 Detailed Description

Header which includes classes to convert a parsed TwinCAT tpy into an EPICS database. It includes "ParseTpy.h"

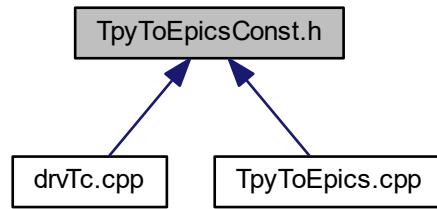
9.30 TpyToEpicsConst.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for TpyToEpicsConst.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseTpy](#)
Namespace for parsing.
- [EpicsTpy](#)
Namespace for tpy-db conversion.

Variables

- const int [EpicsTpy::MAX_EPICS_CHANNEL](#) = 54
- const int [EpicsTpy::MAX_EPICS_DESC](#) = 40
- const int [EpicsTpy::MAX_EPICS_STRING](#) = 40
- const int [EpicsTpy::MAX_EPICS_UNIT](#) = 15
- const char *const [EpicsTpy::EPICS_DB_EGU](#) = "EGU"
- const char *const [EpicsTpy::EPICS_DB_DESC](#) = "DESC"
- const char *const [EpicsTpy::EPICS_DB_HOPR](#) = "HOPR"
- const char *const [EpicsTpy::EPICS_DB_LOPR](#) = "LOPR"
- const char *const [EpicsTpy::EPICS_DB_DRVH](#) = "DRVH"
- const char *const [EpicsTpy::EPICS_DB_DRVL](#) = "DRVL"
- const char *const [EpicsTpy::EPICS_DB_ONAM](#) = "ONAM"
- const char *const [EpicsTpy::EPICS_DB_ZNAM](#) = "ZNAM"
- const char *const [EpicsTpy::EPICS_DB_PREC](#) = "PREC"
- const char *const [EpicsTpy::EPICS_DB_ZRST](#) [16]

- const char *const EpicsTpy::EPICS_DB_ZRVL [16]
- const char *const EpicsTpy::EPICS_DB_SCAN = "SCAN"
- const char *const EpicsTpy::EPICS_DB_INP = "INP"
- const char *const EpicsTpy::EPICS_DB_OUT = "OUT"
- const char *const EpicsTpy::EPICS_DB_TSE = "TSE"
- const char *const EpicsTpy::EPICS_DB_PINI = "PINI"
- const char *const EpicsTpy::EPICS_DB_DTYP = "DTYP"
- const char *const EpicsTpy::EPICS_DB_OSV = "OSV"
- const char *const EpicsTpy::EPICS_DB_ZSV = "ZSV"
- const char *const EpicsTpy::EPICS_DB_COSV = "COSV"
- const char *const EpicsTpy::EPICS_DB_HIHI = "HIHI"
- const char *const EpicsTpy::EPICS_DB_HIGH = "HIGH"
- const char *const EpicsTpy::EPICS_DB_LOW = "LOW"
- const char *const EpicsTpy::EPICS_DB_LOLO = "LOLO"
- const char *const EpicsTpy::EPICS_DB_HYST = "HYST"
- const char *const EpicsTpy::EPICS_DB_HHSV = "HHSV"
- const char *const EpicsTpy::EPICS_DB_HSV = "HSV"
- const char *const EpicsTpy::EPICS_DB_LSV = "LSV"
- const char *const EpicsTpy::EPICS_DB_LLSV = "LLSV"
- const char *const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"
- const char *const EpicsTpy::EPICS_DB_MINOR = "MINOR"
- const char *const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"
- const char *const EpicsTpy::EPICS_DB_UNSV = "UNSV"
- const char *const EpicsTpy::EPICS_DB_ZRSV [16]
- const char *const EpicsTpy::EPICS_DB_FORBIDDEN []
- const char *const EpicsTpy::EPICS_DB_ALLOWED []
- const char *const EpicsTpy::EPICS_DB_NUMVAL []
- const char *const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"
- const int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4
- const int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2
- const int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT
- const char *const EpicsTpy::LIGODAQ_UNIT_NAME = "units"
- const char *const EpicsTpy::LIGODAQ_UNIT_NONE = "none"
- const char *const EpicsTpy::LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE
- const char *const EpicsTpy::LIGODAQ_INI_HEADER

9.30.1 Detailed Description

Header which includes EPICS constants needed to generate an EPICS db file from a parsed TwinCAT tpy file.

Index

aaival
 Device support for TwinCAT/ADS, [17](#)

aooval
 Device support for TwinCAT/ADS, [17](#)

access_rights_enum
 plc, [94](#)

add
 plc::BasePLC, [118](#)
 plc::System, [289](#)
 TcComms::tcProcWrite, [307](#)

ADSCallback
 TcComms, [101](#)
 TcComms::TcPLC, [305](#)

airval
 Device support for TwinCAT/ADS, [17](#)

aival
 Device support for TwinCAT/ADS, [17](#)

all
 EPICS utility functions and classes, [60](#)

aorval
 Device support for TwinCAT/ADS, [17](#)

aoaval
 Device support for TwinCAT/ADS, [17](#)

arraytype
 TwinCAT tpy file parser, [27](#)

atomic_string.h, [325](#)

base_record
 ParseTpy::base_record, [114](#), [115](#)

BasePLCList
 plc, [93](#)

BasePLCPtr
 plc, [93](#)

BaseRecord
 plc::BaseRecord, [129](#)

BaseRecordList
 plc, [93](#)

BaseRecordPtr
 plc, [93](#)

birval
 Device support for TwinCAT/ADS, [17](#)

bival
 Device support for TwinCAT/ADS, [17](#)

borval
 Device support for TwinCAT/ADS, [17](#)

boval
 Device support for TwinCAT/ADS, [17](#)

case_type
 EPICS utility functions and classes, [58](#)

closed
 EPICS utility functions and classes, [59](#)

closePort
 TcComms::TcPLC, [301](#)

compare
 String functions and classes, [49](#)

compareNamesWoNamespace
 ParseTpy, [88](#)

cyclesLeft
 TcComms::TcPLC, [306](#)

data_enum
 plc::DataValueTraits< T >, [188](#)

data_type_enum
 plc, [94](#)

DataValue
 plc::DataValue, [156](#)

dbinfo_list
 InfoPlc::InfoInterface, [246](#)

Device driver functions, [22](#)

Device support for TwinCAT/ADS, [13](#)
 aaival, [17](#)
 aooval, [17](#)
 airval, [17](#)
 aival, [17](#)
 aorval, [17](#)
 aoaval, [17](#)
 birval, [17](#)
 bival, [17](#)
 borval, [17](#)
 boval, [17](#)
 epics_record_enum, [16](#)
 epics_record_enumEnd, [17](#)
 eventval, [17](#)
 get_callback_queue_free, [17](#)
 get_callback_queue_size, [18](#)
 get_callback_queue_used, [18](#)
 histogramval, [17](#)
 invalidval, [17](#)
 isCallback, [20](#)
 isPassive, [20](#)
 linkRecord, [19](#)
 longinval, [17](#)
 longoutval, [17](#)
 mbbiDirectrv, [17](#)
 mbbiDirectval, [17](#)
 mbbirval, [17](#)
 mbbival, [17](#)
 mbboDirectrv, [17](#)
 mbboDirectval, [17](#)

mbborval, 17
 mbboval, 17
 register_devsup, 19
 stringintval, 17
 stringoutval, 17
 the_register_devsup, 20
 value_ait_type, 21
 value_count, 21
 waveformval, 17
 device_support_opc_name
 EPICS utility functions and classes, 59
 device_support_tc_name
 EPICS utility functions and classes, 59
 device_support_type
 EPICS utility functions and classes, 58
 DevTc, 75
 tc_regex, 76
 tcAlias, 77
 tcInfoPrefix, 78
 tcList, 78
 tcLoadRecords, 79
 tcMacro, 79
 tcPrintVal, 80
 tcPrintVals, 80
 tcSetScanRate, 82
 devTc.cpp, 326
 devTc.h, 328
 DevTc::devTcDefIn< RecType >, 190
 DevTc::devTcDeflo< RecType >, 192
 DevTc::devTcDefOut< RecType >, 194
 DevTc::devTcDefWaveformIn< RecType >, 195
 DevTc::epics_record_traits< RecType >, 228
 DevTc::epics_record_traits< RecType >::traits_type, 320
 DevTc::epics_tc_db_processing, 229
 epics_tc_db_processing, 231
 operator(), 231
 patch_db_recordnames, 232
 process_list, 232
 process_lists, 233
 process_macro, 233
 process_macros, 233
 DevTc::EpicsInterface, 234
 DevTc::register_devsup, 276
 DevTc::tcRegisterTolocShell, 309
 dimension
 TwinCAT tpy file parser, 25
 dimensions
 TwinCAT tpy file parser, 26
 done
 Info interface classes and functions, 24
 drvTc.cpp, 330
 drvTc.h, 332
 dtBinary
 plc, 95
 dtBool
 plc, 95
 dtDouble
 plc, 95
 dtFloat
 plc, 95
 dtInt16
 plc, 95
 dtInt32
 plc, 95
 dtInt64
 plc, 95
 dtInt8
 plc, 95
 dtInvalid
 plc, 95
 dtString
 plc, 95
 dtUInt16
 plc, 95
 dtUInt32
 plc, 95
 dtUInt64
 plc, 95
 dtUInt8
 plc, 95
 dtWString
 plc, 95
 enum_map
 TwinCAT tpy file parser, 26
 enum_pair
 TwinCAT tpy file parser, 26
 enumtype
 TwinCAT tpy file parser, 27
 EPICS constants, 61
 EPICS_DB_ALLOWED, 62
 EPICS_DB_COSV, 62
 EPICS_DB_DESC, 62
 EPICS_DB_DRVH, 62
 EPICS_DB_DRVL, 63
 EPICS_DB_DTYP, 63
 EPICS_DB_EGU, 63
 EPICS_DB_FORBIDDEN, 63
 EPICS_DB_HHSV, 64
 EPICS_DB_HIGH, 64
 EPICS_DB_HIHI, 64
 EPICS_DB_HOPR, 64
 EPICS_DB_HSV, 65
 EPICS_DB_HYST, 65
 EPICS_DB_INP, 65
 EPICS_DB_LLSV, 65
 EPICS_DB_LOLO, 66
 EPICS_DB_LOPR, 66
 EPICS_DB_LOW, 66
 EPICS_DB_LSV, 66
 EPICS_DB_MAJOR, 67
 EPICS_DB_MINOR, 67
 EPICS_DB_NOALARM, 67
 EPICS_DB_NUMVAL, 67
 EPICS_DB_ONAM, 68
 EPICS_DB_OSV, 68

EPICS_DB_OUT, 68
EPICS_DB_PINI, 68
EPICS_DB_PREC, 69
EPICS_DB_SCAN, 69
EPICS_DB_TSE, 69
EPICS_DB_UNSV, 69
EPICS_DB_ZNAM, 70
EPICS_DB_ZRST, 70
EPICS_DB_ZRSV, 70
EPICS_DB_ZRVL, 70
EPICS_DB_ZSV, 71
LIGODAQ_DATATYPE_DEFAULT, 71
LIGODAQ_DATATYPE_FLOAT, 71
LIGODAQ_DATATYPE_INT32, 71
LIGODAQ_DATATYPE_NAME, 72
LIGODAQ_INI_HEADER, 72
LIGODAQ_UNIT_DEFAULT, 72
LIGODAQ_UNIT_NAME, 72
LIGODAQ_UNIT_NONE, 73
MAX_EPICS_CHANNEL, 73
MAX_EPICS_DESC, 73
MAX_EPICS_STRING, 73
MAX_EPICS_UNIT, 73
EPICS utility functions and classes, 57
 all, 60
 case_type, 58
 closed, 59
 device_support_opc_name, 59
 device_support_tc_name, 59
 device_support_type, 58
 errors, 60
 fields, 60
 filename_set, 57
 io_filestat, 59
 ligo_std, 60
 ligo_vac, 60
 listing_autoburt, 59
 listing_daqini, 59
 listing_standard, 59
 listing_type, 59
 lower_case, 58
 macro_list, 58
 macro_stack, 58
 macrofile_type, 59
 no_conversion, 60
 no_dot, 60
 preserve_case, 58
 tc_epics_conv, 60
 upper_case, 58
 epics_conversion
 EpicsTpy::epics_conversion, 198, 199
EPICS_DB_ALLOWED
 EPICS constants, 62
EPICS_DB_COSV
 EPICS constants, 62
EPICS_DB_DESC
 EPICS constants, 62
EPICS_DB_DRVH
 EPICS constants, 62
EPICS_DB_DRLV
 EPICS constants, 63
EPICS_DB_DTYP
 EPICS constants, 63
EPICS_DB_EGU
 EPICS constants, 63
EPICS_DB_FORBIDDEN
 EPICS constants, 63
EPICS_DB_HHSV
 EPICS constants, 64
EPICS_DB_HIGH
 EPICS constants, 64
EPICS_DB_HIHI
 EPICS constants, 64
EPICS_DB_HOPR
 EPICS constants, 64
EPICS_DB_HSV
 EPICS constants, 65
EPICS_DB_HYST
 EPICS constants, 65
EPICS_DB_INP
 EPICS constants, 65
EPICS_DB_LLSV
 EPICS constants, 65
EPICS_DB_LOLO
 EPICS constants, 66
EPICS_DB_LOPR
 EPICS constants, 66
EPICS_DB_LOW
 EPICS constants, 66
EPICS_DB_LSV
 EPICS constants, 66
EPICS_DB_MAJOR
 EPICS constants, 67
EPICS_DB_MINOR
 EPICS constants, 67
EPICS_DB_NUMVAL
 EPICS constants, 67
EPICS_DB_ONAM
 EPICS constants, 68
EPICS_DB_OSV
 EPICS constants, 68
EPICS_DB_OUT
 EPICS constants, 68
EPICS_DB_PINI
 EPICS constants, 68
EPICS_DB_PREC
 EPICS constants, 69
epics_db_processing
 EpicsTpy::epics_db_processing, 204
EPICS_DB_SCAN
 EPICS constants, 69
EPICS_DB_TSE
 EPICS constants, 69
EPICS_DB_UNSV

EPICS constants, 69
EPICS_DB_ZNAM
 EPICS constants, 70
EPICS_DB_ZRST
 EPICS constants, 70
EPICS_DB_ZRSV
 EPICS constants, 70
EPICS_DB_ZRVL
 EPICS constants, 70
EPICS_DB_ZSV
 EPICS constants, 71
epics_list_processing
 EpicsTpy::epics_list_processing, 215, 216
epics_macrofiles_processing
 EpicsTpy::epics_macrofiles_processing, 223
epics_record_enum
 Device support for TwinCAT/ADS, 16
epics_record_enumEnd
 Device support for TwinCAT/ADS, 17
epics_tc_db_processing
 DevTc::epics_tc_db_processing, 231
EpicsDbGen.cpp, 333
 main, 333
EpicsTpy, 83
EpicsTpy::epics_conversion, 196
 epics_conversion, 198, 199
 getopt, 199
 to_epics, 201
EpicsTpy::epics_db_processing, 202
 epics_db_processing, 204
 getopt, 205
 my getopt, 206
 operator(), 207
 process_field_alarm, 209
 process_field_numeric, 210–212
 process_field_string, 212
EpicsTpy::epics_list_processing, 214
 epics_list_processing, 215, 216
 getopt, 216
 my getopt, 217
 operator(), 218
EpicsTpy::epics_macrofiles_processing, 220
 epics_macrofiles_processing, 223
 getopt, 224
 my getopt, 225
 operator(), 226
 to_filename, 227
EpicsTpy::macro_info, 250
EpicsTpy::macro_record, 251
EpicsTpy::multi_io_support, 256
 getopt, 259
 multi_io_support, 258
EpicsTpy::split_io_support, 280
 getopt, 283
 increment, 284
 operator=, 285
 split_io_support, 282, 283
eq
String functions and classes, 50
erase
 plc::BasePLC, 120
errorPrintf
 TcComms, 101
errors
 EPICS utility functions and classes, 60
eventval
 Device support for TwinCAT/ADS, 17
fields
 EPICS utility functions and classes, 60
filename_set
 EPICS utility functions and classes, 57
find
 plc::BasePLC, 120
for_each
 plc::BasePLC, 121
 plc::System, 289
forever
 Info interface classes and functions, 24
functionblock
 TwinCAT tpy file parser, 27
get
 ParseTpy::ads_routing_info, 104, 105
 ParseUtil::memory_location, 254
 ParseUtil::process_arg_tc, 274
get_callback_queue_free
 Device support for TwinCAT/ADS, 17
get_callback_queue_size
 Device support for TwinCAT/ADS, 18
get_callback_queue_used
 Device support for TwinCAT/ADS, 18
get_decoration
 ParseTpy, 88
get_full
 InfoPlc::process_arg_info, 271
 ParseUtil::process_arg, 269
 ParseUtil::process_arg_tc, 274
get_infodb
 InfoPlc::InfoInterface, 244
get_next
 plc::BasePLC, 121
get_pointer
 ParseTpy, 89
get_responseBuffer
 TcComms::TcPLC, 302
get_symbol_name
 InfoPlc::InfoInterface, 245
get_timestamp_unix
 plc::BasePLC, 122
getopt
 EpicsTpy::epics_conversion, 199
 EpicsTpy::epics_db_processing, 205
 EpicsTpy::epics_list_processing, 216
 EpicsTpy::epics_macrofiles_processing, 224
 EpicsTpy::multi_io_support, 259
 EpicsTpy::split_io_support, 283

ParseUtil::tag_processing, 293
GetValid
 plc::DataValue, 157

histogramval
 Device support for TwinCAT/ADS, 17

increment
 EpicsTpy::split_io_support, 284

Info interface classes and functions, 23
 done, 24
 forever, 24
 info_dbrecord_type, 23
 once, 24
 update_enum, 23

info_dbrecord_type
 Info interface classes and functions, 23

InfoInterface
 InfoPlc::InfoInterface, 243

InfoPlc, 85
infoPlc.cpp, 334
infoPlc.h, 335
InfoPlc::InfoInterface, 239
 dbinfo_list, 246
 get_infodb, 244
 get_symbol_name, 245
 InfoInterface, 243
 printVal, 245

InfoPlc::process_arg_info, 270
 get_full, 271
 process_arg_info, 271

infoPlcTemplate.h, 336

Init
 plc::DataValue, 157

Interface
 plc::Interface, 248

InterfacePtr
 plc, 94

invalidval
 Device support for TwinCAT/ADS, 17

io_filestat
 EPICS utility functions and classes, 59

iocMain.cpp, 337
 main, 338

isCallback
 Device support for TwinCAT/ADS, 20

isPassive
 Device support for TwinCAT/ADS, 20

item_list
 TwinCAT tpy file parser, 26

ligo_std
 EPICS utility functions and classes, 60

ligo_vac
 EPICS utility functions and classes, 60

LIGODAQ_DATATYPE_DEFAULT
 EPICS constants, 71

LIGODAQ_DATATYPE_FLOAT
 EPICS constants, 71

LIGODAQ_DATATYPE_INT32
 EPICS constants, 71

LIGODAQ_DATATYPE_NAME
 EPICS constants, 72

LIGODAQ_INI_HEADER
 EPICS constants, 72

LIGODAQ_UNIT_DEFAULT
 EPICS constants, 72

LIGODAQ_UNIT_NAME
 EPICS constants, 72

LIGODAQ_UNIT_NONE
 EPICS constants, 73

linkRecord
 Device support for TwinCAT/ADS, 19

listing_autoburt
 EPICS utility functions and classes, 59

listing_daqini
 EPICS utility functions and classes, 59

listing_standard
 EPICS utility functions and classes, 59

listing_type
 EPICS utility functions and classes, 59

longinval
 Device support for TwinCAT/ADS, 17

longoutval
 Device support for TwinCAT/ADS, 17

lower_case
 EPICS utility functions and classes, 58

lt
 String functions and classes, 51

macro_list
 EPICS utility functions and classes, 58

macro_stack
 EPICS utility functions and classes, 58

macrofile_type
 EPICS utility functions and classes, 59

main
 EpicsDbGen.cpp, 333
 iocMain.cpp, 338
 ParseTpyInfo.cpp, 344

MAX_EPICS_CHANNEL
 EPICS constants, 73

MAX_EPICS_DESC
 EPICS constants, 73

MAX_EPICS_STRING
 EPICS constants, 73

MAX_EPICS_UNIT
 EPICS constants, 73

mbbiDirectrv
 Device support for TwinCAT/ADS, 17

mbbiDirectval
 Device support for TwinCAT/ADS, 17

mbbirval
 Device support for TwinCAT/ADS, 17

mbbival
 Device support for TwinCAT/ADS, 17

mbboDirectrv
 Device support for TwinCAT/ADS, 17

mbboDirectval
 Device support for TwinCAT/ADS, 17
mbborval
 Device support for TwinCAT/ADS, 17
mbboval
 Device support for TwinCAT/ADS, 17
memory_location
 ParseUtil::memory_location, 253, 254
multi_io_support
 EpicsTpy::multi_io_support, 258
mygetopt
 EpicsTpy::epics_db_processing, 206
 EpicsTpy::epics_list_processing, 217
 EpicsTpy::epics_macrofiles_processing, 225
mysize
 plc::DataValue, 186

name_parse
 ParseTpy::parserinfo_type, 266
ne
 String functions and classes, 51, 52
no_change
 Parser utility functions and classes, 31
no_conversion
 EPICS utility functions and classes, 60
no_dot
 EPICS utility functions and classes, 60

once
 Info interface classes and functions, 24
OPC property constants, 33
 OPC_PROP_ALIAS, 34
 OPC_PROP_ALMAREAS, 34
 OPC_PROP_ALMCONDITION, 34
 OPC_PROP_ALMCOSV, 34
 OPC_PROP_ALMDB, 35
 OPC_PROP_ALMDEV, 35
 OPC_PROP_ALMFFSV, 35
 OPC_PROP_ALMH, 35
 OPC_PROP_ALMHELP, 36
 OPC_PROP_ALMH, 36
 OPC_PROP_ALMHHSV, 36
 OPC_PROP_ALMHHSV, 36
 OPC_PROP_ALML, 37
 OPC_PROP_ALMLIMIT, 37
 OPC_PROP_ALMLL, 37
 OPC_PROP_ALMLLSV, 37
 OPC_PROP_ALMLSV, 38
 OPC_PROP_ALMOSV, 38
 OPC_PROP_ALMPRIMARYAREA, 38
 OPC_PROP_ALMROC, 38
 OPC_PROP_ALMSTAT, 38
 OPC_PROP_ALMUNSV, 39
 OPC_PROP_ALMZRSV, 39
 OPC_PROP_ALMZSV, 39
 OPC_PROP_AVI, 39
 OPC_PROP_BGC, 39
 OPC_PROP_BLINK, 40
 OPC_PROP_BMP, 40

 OPC_PROP_CDT, 40
 OPC_PROP_CLOSE, 40
 OPC_PROP_DESC, 40
 OPC_PROP_DTYP, 41
 OPC_PROP_FFST, 41
 OPC_PROP_FGC, 41
 OPC_PROP_HIEU, 41
 OPC_PROP_HIRANGE, 42
 OPC_PROP_HTML, 42
 OPC_PROP_INOUT, 42
 OPC_PROP_INPUT, 42
 OPC_PROP_LOEU, 43
 OPC_PROP_LORANGE, 43
 OPC_PROP_OPEN, 43
 OPC_PROP_OUTPUT, 43
 OPC_PROP_PINI, 44
 OPC_PROP_PLNAME, 44
 OPC_PROP_PREC, 44
 OPC_PROP_QUALITY, 44
 OPC_PROP_RECTYPE, 45
 OPC_PROP_RIGHTS, 45
 OPC_PROP_SCANRATE, 45
 OPC_PROP_SERVER, 45
 OPC_PROP SND, 45
 OPC_PROP_TIME, 46
 OPC_PROP_TIMEZONE, 46
 OPC_PROP_TSE, 46
 OPC_PROP_UNIT, 46
 OPC_PROP_VALUE, 46
 OPC_PROP_ZRST, 47
opc_enum
 Parser utility functions and classes, 31
OPC_PROP_ALIAS
 OPC property constants, 34
OPC_PROP_ALMAREAS
 OPC property constants, 34
OPC_PROP_ALMCONDITION
 OPC property constants, 34
OPC_PROP_ALMCOSV
 OPC property constants, 34
OPC_PROP_ALMDB
 OPC property constants, 35
OPC_PROP_ALMDEV
 OPC property constants, 35
OPC_PROP_ALMFFSV
 OPC property constants, 35
OPC_PROP_ALMH
 OPC property constants, 35
OPC_PROP_ALMHELP
 OPC property constants, 36
OPC_PROP_ALMH
 OPC property constants, 36
OPC_PROP_ALMHHSV
 OPC property constants, 36
OPC_PROP_ALMHHSV
 OPC property constants, 36
OPC_PROP_ALML
 OPC property constants, 37

OPC_PROP_ALMLIMIT
 OPC property constants, 37

OPC_PROP_ALMLL
 OPC property constants, 37

OPC_PROP_ALMLLSV
 OPC property constants, 37

OPC_PROP_ALMLSV
 OPC property constants, 38

OPC_PROP_ALMOSV
 OPC property constants, 38

OPC_PROP_ALMPRIMARYAREA
 OPC property constants, 38

OPC_PROP_ALMROC
 OPC property constants, 38

OPC_PROP_ALMSTAT
 OPC property constants, 38

OPC_PROP_ALMUNSV
 OPC property constants, 39

OPC_PROP_ALMZRSV
 OPC property constants, 39

OPC_PROP_ALMZSV
 OPC property constants, 39

OPC_PROP_AVI
 OPC property constants, 39

OPC_PROP_BGC
 OPC property constants, 39

OPC_PROP_BLINK
 OPC property constants, 40

OPC_PROP_BMP
 OPC property constants, 40

OPC_PROP_CDT
 OPC property constants, 40

OPC_PROP_CLOSE
 OPC property constants, 40

OPC_PROP_DESC
 OPC property constants, 40

OPC_PROP_DTYP
 OPC property constants, 41

OPC_PROP_FFST
 OPC property constants, 41

OPC_PROP_FGC
 OPC property constants, 41

OPC_PROP_HIEU
 OPC property constants, 41

OPC_PROP_HIRANGE
 OPC property constants, 42

OPC_PROP_HTML
 OPC property constants, 42

OPC_PROP_INOUT
 OPC property constants, 42

OPC_PROP_INPUT
 OPC property constants, 42

OPC_PROP_LOEU
 OPC property constants, 43

OPC_PROP_LORANGE
 OPC property constants, 43

OPC_PROP_OPEN
 OPC property constants, 43

OPC_PROP_OUTPUT
 OPC property constants, 43

OPC_PROP_PINI
 OPC property constants, 44

OPC_PROP_PLNAME
 OPC property constants, 44

OPC_PROP_PREC
 OPC property constants, 44

OPC_PROP_QUALITY
 OPC property constants, 44

OPC_PROP_RECTYPE
 OPC property constants, 45

OPC_PROP_RIGHTS
 OPC property constants, 45

OPC_PROP_SCANRATE
 OPC property constants, 45

OPC_PROP_SERVER
 OPC property constants, 45

OPC_PROP SND
 OPC property constants, 45

OPC_PROP_TIME
 OPC property constants, 46

OPC_PROP_TIMEZONE
 OPC property constants, 46

OPC_PROP_TSE
 OPC property constants, 46

OPC_PROP_UNIT
 OPC property constants, 46

OPC_PROP_VALUE
 OPC property constants, 46

OPC_PROP_ZRST
 OPC property constants, 47

operator()
 DevTc::epics_tc_db_processing, 231
 EpicsTpy::epics_db_processing, 207
 EpicsTpy::epics_list_processing, 218
 EpicsTpy::epics_macrofiles_processing, 226
 std::std::hash< std::stringcase >, 237
 std::std::hash< std::wstringcase >, 238

operator=
 EpicsTpy::split_io_support, 285
 std::atomic_string< stringT >, 112

optarg
 ParseUtil::optarg, 262

optimizeRequests
 TcComms::TcPLC, 302

parse
 ParseUtil::optarg, 263

parse_finish
 ParseTpy::tpy_file, 312

Parser utility functions and classes, 30
 no_change, 31
 opc_enum, 31
 process_all, 32
 process_atomic, 32
 process_structured, 32
 process_tag_enum, 31
 process_type_enum, 32

property_el, 30
 property_map, 31
 pt_binary, 32
 pt_bool, 32
 pt_enum, 32
 pt_int, 32
 pt_invalid, 32
 pt_real, 32
 pt_string, 32
 publish, 31
 replacement_table, 31
 silent, 31
ParseTpy, 85
 compareNamesWoNamespace, 88
 get_decoration, 88
 get_pointer, 89
ParseTpy.cpp, 338
ParseTpy.h, 339
ParseTpy::ads_routing_info, 103
 get, 104, 105
 set, 105
ParseTpy::base_record, 112
 base_record, 114, 115
ParseTpy::compiler_info, 151
ParseTpy::item_record, 249
ParseTpy::parserinfo_type, 264
 name_parse, 266
ParseTpy::project_record, 275
ParseTpy::symbol_record, 286
ParseTpy::tpy_file, 310
 parse_finish, 312
 process_array, 312
 process_symbols, 313
 process_type_tree, 314, 316, 318
ParseTpy::type_map, 320
ParseTpy::type_record, 321
ParseTpyConst.h, 341
ParseTpyInfo.cpp, 343
 main, 344
ParseTpyTemplate.h, 344
ParseUtil, 89
ParseUtil.cpp, 345
ParseUtil.h, 345
ParseUtil::bit_location, 147
ParseUtil::memory_location, 252
 get, 254
 memory_location, 253, 254
 set, 255
 set_section, 255
ParseUtil::opc_list, 260
ParseUtil::optarg, 261
 optarg, 262
 parse, 263
ParseUtil::process_arg, 267
 get_full, 269
 process_arg, 268
ParseUtil::process_arg_tc, 272
 get, 274
 get_full, 274
 process_arg_tc, 273
ParseUtil::replacement_rules, 278
ParseUtil::tag_processing, 291
 getopt, 293
 tag_processing, 292, 293
ParseUtil::variable_name, 323
ParseUtilConst.h, 347
patch_db_recordnames
 DevTc::epics_tc_db_processing, 232
plc, 91
 access_rights_enum, 94
BasePLCList, 93
BasePLCPtr, 93
BaseRecordList, 93
BaseRecordPtr, 93
 data_type_enum, 94
 dtBinary, 95
 dtBool, 95
 dtDouble, 95
 dtFloat, 95
 dtInt16, 95
 dtInt32, 95
 dtInt64, 95
 dtInt8, 95
 dtInvalid, 95
 dtString, 95
 dtUInt16, 95
 dtUInt32, 95
 dtUInt64, 95
 dtUInt8, 95
 dtWString, 95
 InterfacePtr, 94
 read_only, 94
 read_write, 94
 reset_and_read, 95
 ScannerProc, 96
 scannerThread, 97
 write_and_test, 98, 99
 write_only, 94
plc::BasePLC, 115
 add, 118
 erase, 120
 find, 120
 for_each, 121
 get_next, 121
 get_timestamp_unix, 122
 plc_data_set_valid, 123
 printRecord, 124
 records, 125
 reserve, 124
 start, 124
 user_data_set_valid, 124
plc::BaseRecord, 126
 BaseRecord, 129
 PlcGetValid, 129
 PlcPush, 130
 PlcRead, 131, 132

PlcReadBinary, 133
PlcSetValid, 134
PlcWrite, 135, 136
PlcWriteBinary, 137
UserGetValid, 138
UserPush, 138
UserRead, 140–142
UserReadBinary, 142
UserSetValid, 143
UserWrite, 144–146
UserWriteBinary, 146
plc::DataValue, 153
 WithValue, 156
 GetValid, 157
 Init, 157
 mysize, 186
 PlcGetValid, 158
 PlcRead, 159–161
 PlcReadBinary, 161
 PlcSetValid, 162
 PlcWrite, 163–165
 PlcWriteBinary, 166
 Read, 167–169
 ReadBinary, 170
 SetValid, 171
 UserGetValid, 172
 UserRead, 172–174
 UserReadBinary, 175
 UserSetValid, 176
 UserWrite, 176–178
 UserWriteBinary, 179
 Write, 180–183
 WriteBinary, 185
plc::DataValueTraits< T >, 186
 data_enum, 188
plc::DataValueTypeDef, 188
plc::Interface, 246
 Interface, 248
 printVal, 248
plc::scanner_thread_args, 279
plc::System, 288
 add, 289
 for_each, 289
 printVal, 290
plc_data_set_valid
 plc::BasePLC, 123
plcBase.cpp, 349
plcBase.h, 350
plcBaseTemplate.h, 352
PlcGetValid
 plc::BaseRecord, 129
 plc::DataValue, 158
PlcPush
 plc::BaseRecord, 130
PlcRead
 plc::BaseRecord, 131, 132
 plc::DataValue, 159–161
PlcReadBinary
 plc::BaseRecord, 133
 plc::DataValue, 161
PlcSetValid
 plc::BaseRecord, 134
 plc::DataValue, 162
PlcWrite
 plc::BaseRecord, 135, 136
 plc::DataValue, 163–165
PlcWriteBinary
 plc::BaseRecord, 137
 plc::DataValue, 166
preserve_case
 EPICS utility functions and classes, 58
printRecord
 plc::BasePLC, 124
 TcComms::TcPLC, 303
printVal
 InfoPlc::InfoInterface, 245
 plc::Interface, 248
 plc::System, 290
 TcComms::TCatInterface, 297
process_all
 Parser utility functions and classes, 32
process_arg
 ParseUtil::process_arg, 268
process_arg_info
 InfoPlc::process_arg_info, 271
process_arg_tc
 ParseUtil::process_arg_tc, 273
process_array
 ParseTpy::tpy_file, 312
process_atomic
 Parser utility functions and classes, 32
process_field_alarm
 EpicsTpy::epics_db_processing, 209
process_field_numeric
 EpicsTpy::epics_db_processing, 210–212
process_field_string
 EpicsTpy::epics_db_processing, 212
process_list
 DevTc::epics_tc_db_processing, 232
process_lists
 DevTc::epics_tc_db_processing, 233
process_macro
 DevTc::epics_tc_db_processing, 233
process_macros
 DevTc::epics_tc_db_processing, 233
process_structured
 Parser utility functions and classes, 32
process_symbols
 ParseTpy::tpy_file, 313
process_tag_enum
 Parser utility functions and classes, 31
process_type_enum
 Parser utility functions and classes, 32
process_type_tree
 ParseTpy::tpy_file, 314, 316, 318
property_el

Parser utility functions and classes, 30
property_map
 Parser utility functions and classes, 31
pt_binary
 Parser utility functions and classes, 32
pt_bool
 Parser utility functions and classes, 32
pt_enum
 Parser utility functions and classes, 32
pt_int
 Parser utility functions and classes, 32
pt_invalid
 Parser utility functions and classes, 32
pt_real
 Parser utility functions and classes, 32
pt_string
 Parser utility functions and classes, 32
publish
 Parser utility functions and classes, 31

Read
 plc::DataValue, 167–169
read_only
 plc, 94
read_ptr
 TcComms::tcProcWrite, 308
read_write
 plc, 94
ReadBinary
 plc::DataValue, 170
records
 plc::BasePLC, 125
register_devsup
 Device support for TwinCAT/ADS, 19
replacement_table
 Parser utility functions and classes, 31
req
 TcComms::tcProcWrite, 309
reserve
 plc::BasePLC, 124
reset_and_read
 plc, 95
RouterCall
 TcComms, 102
 TcComms::AmsRouterNotification, 107

ScannerProc
 plc, 96
scannerThread
 plc, 97
set
 ParseTpy::ads_routing_info, 105
 ParseUtil::memory_location, 255
set_addr
 TcComms::TcPLC, 303
set_ads_state
 TcComms::TcPLC, 304
set_section
 ParseUtil::memory_location, 255

SetValid
 plc::DataValue, 171
silent
 Parser utility functions and classes, 31
simple
 TwinCAT tpy file parser, 27
split_io_support
 EpicsTpy::split_io_support, 282, 283
split_string
 String functions and classes, 52
start
 plc::BasePLC, 124
 std::atomic< string >, 107
 std::atomic< wstring >, 109
 std::atomic_string< stringT >, 110
 operator=, 112
 std::case_char_traits, 148
 std::case_wchar_traits, 150
 std::std::hash< std::stringcase >, 237
 operator(), 237
 std::std::hash< std::wstringcase >, 238
 operator(), 238
 stdafx.cpp, 353
 stdafx.h, 353
String functions and classes, 48
 compare, 49
 eq, 50
 lt, 51
 ne, 51, 52
 split_string, 52
 stringcase, 48
 strncasecmp, 53
 trim_space, 53, 54
 wcsncasecmp, 54
 wstringcase, 48
stringcase
 String functions and classes, 48
stringcase.h, 354
stringcase_hash.h, 355
stringintval
 Device support for TwinCAT/ADS, 17
stringoutval
 Device support for TwinCAT/ADS, 17
strncasecmp
 String functions and classes, 53
structtype
 TwinCAT tpy file parser, 27
symbol_list
 TwinCAT tpy file parser, 26
syminfo_processing, 287

tag_processing
 ParseUtil::tag_processing, 292, 293
tc_epics_conv
 EPICS utility functions and classes, 60
tc_regex
 DevTc, 76
tcAlias
 DevTc, 77

TCatInterface
 TcComms::TCatInterface, 297

TcComms, 99
 ADScallback, 101
 errorPrintf, 101
 RouterCall, 102

tcComms.cpp, 356

tcComms.h, 357

TcComms::AmsRouterNotification, 106
 RouterCall, 107

TcComms::DataPar, 152

TcComms::TCatInterface, 294
 printVal, 297
 TCatInterface, 297

TcComms::TcPLC, 298
 ADScallback, 305
 closePort, 301
 cyclesLeft, 306
 get_responseBuffer, 302
 optimizeRequests, 302
 printRecord, 303
 set_addr, 303
 set_ads_state, 304
 update_scanner, 304

TcComms::tcProcWrite, 306
 add, 307
 read_ptr, 308
 req, 309

tclInfoPrefix
 DevTc, 78

tcList
 DevTc, 78

tcLoadRecords
 DevTc, 79

tcMacro
 DevTc, 79

tcPrintVal
 DevTc, 80

tcPrintVals
 DevTc, 80

tcSetScanRate
 DevTc, 82

the_register_devsup
 Device support for TwinCAT/ADS, 20

to_epics
 EpicsTpy::epics_conversion, 201

to_filename
 EpicsTpy::epics_macrofiles_processing, 227

TpyToEpics.cpp, 358

TpyToEpics.h, 359

TpyToEpicsConst.h, 361

trim_space
 String functions and classes, 53, 54

TwinCAT read/write scanning, 56

TwinCAT tpy file parser, 25
 arraytype, 27
 dimension, 25
 dimensions, 26

 enum_map, 26
 enum_pair, 26
 enumtype, 27
 functionblock, 27
 item_list, 26
 simple, 27
 structtype, 27
 symbol_list, 26
 type_enum, 27
 type_multipmap, 27
 unknown, 27

 type_enum
 TwinCAT tpy file parser, 27

 type_multipmap
 TwinCAT tpy file parser, 27

 unknown
 TwinCAT tpy file parser, 27

 update_enum
 Info interface classes and functions, 23

 update_scanner
 TcComms::TcPLC, 304

 upper_case
 EPICS utility functions and classes, 58

 user_data_set_valid
 plc::BasePLC, 124

 UserGetValid
 plc::BaseRecord, 138
 plc::DataValue, 172

 UserPush
 plc::BaseRecord, 138

 UserRead
 plc::BaseRecord, 140–142
 plc::DataValue, 172–174

 UserReadBinary
 plc::BaseRecord, 142
 plc::DataValue, 175

 UserSetValid
 plc::BaseRecord, 143
 plc::DataValue, 176

 UserWrite
 plc::BaseRecord, 144–146
 plc::DataValue, 176–178

 UserWriteBinary
 plc::BaseRecord, 146
 plc::DataValue, 179

 value_ait_type
 Device support for TwinCAT/ADS, 21

 value_count
 Device support for TwinCAT/ADS, 21

 waveformval
 Device support for TwinCAT/ADS, 17

 wcsncasecmp
 String functions and classes, 54

 Write
 plc::DataValue, 180–183

 write_and_test

plc, [98, 99](#)
write_only
 plc, [94](#)
WriteBinary
 plc::DataValue, [185](#)
wstringcase
 String functions and classes, [48](#)

XML tpy file constants, [28](#)